homeLYnk

User Guide

2015 Schneider Electric. All rights reserved.

Version	Description	Created
1.0	Initial release	1.8.2013
A	Updated for firmware 1.2.0	8.4.2014
B	Updated for firmware 1.2.1	15.5.2014
C	Updated for firmware 1.3.0	1.3.2015

2015 Schneider Electric. All rights reserved.

Contents

1 Product Security 10
1.1 Connectivity 10
1.2 Security Features 10
1.3 Operation 10
1.4 Maintenance 10
1.5 Patch Management 11
1.6 Factory Reset 11
1.7 Limitations 11
2 Quick Start Guide 12
2.1 Start Page 13
2.2 Default Configuration 14
2.3 Discover homeLYnk IP Address 16
2.4 Firmware Upgrade 18
2.5 homeLYnk for KNX/EIB Network Configuration Management with ETS3 19
2.6 homeLYnk for KNX/EIB Network Configuration Management with ETS4 20
2.7 homeLYnk for KNX/EIB Network Configuration Management with ETS5 21
2.8 KNX and IP Router Settings 22
2.9 Create Visualization for iPad/PC 23
2.9.1 Import Objects 23
2.9.2 Create ‘Building/Floor’ Structure and Add Objects to the Map 24
2.9.3 Add Objects to Newly Created Visualization Map 26
2.9.4 Launching Visualization on Smartphone Device (iPod in this case) 27
2.9.5 Launching vis. on PC, Tablet or Any Other Touch Device with Large Screen 27
3 Advanced Guide 28
3.1 Configurator’s Main Page 28
3.2 Utilities 30
3.2.1 Import ESF file 30
3.2.2 Import Neighbours 30
3.2.3 Reset / clean-up 31
3.2.4 Factory reset 31
3.2.5 Date and time 32
3.2.6 Install updates 32

2015 Schneider Electric. All rights reserved.

3.2.7 Backup 32
3.2.8 Restore 33
3.2.9 General Configuration 33
3.2.10 Vis.configuration 34
3.3 System 35
3.3.1 System → System 35
3.3.2 System → Network 37
3.3.3 System → Services 44
3.3.4 System → Status 45
3.4 Objects 48
3.4.1 Object parameters 49
3.4.2 Event Script 49
3.4.3 Set object value 50
3.4.4 Object Visualization Parameters 50
3.4.5 Custom Text Value 52
3.4.6 Object Control Bar 52
3.4.7 Object filter 53
3.4.8 Mass edit 53
3.5 Object Logs 54
3.6 Schedulers 55
3.6.1 Add new scheduler 55
3.6.2 Scheduler Events 56
3.6.3 Scheduler Holidays 56
3.7 Trend logs 57
3.7.1 Add New Trend Log 59
3.8 Vis. Structure 60
3.8.1 Levels 60
3.8.2 Second level 60
3.8.3 Plan 61
3.8.4 Layout 62
3.8.5 Widget 63
3.8.6 Visualization Structure Example 64
3.8.7 Plan 64
3.8.8 Layout 65

 2015 Schneider Electric. All rights reserved.

3.8.9 Widget 65
3.8.10 Visualization Object Order 66
3.9 Visualization 67
3.9.1 Structure 68
3.9.2 Visualization Map 69
3.9.3 Plan editor 69
3.9.4 Object 70
3.9.5 Link 73
3.9.6 Camera 74
3.9.7 Graph 75
3.9.8 Text label 76
3.9.9 Image 76
3.9.10 Gauge 77
3.9.11 Frame 78
3.10 Vis. graphics 79
3.11 User access 80
3.12 Scripting 82
3.12.1 Event based 83
3.12.2 Resident 84
3.12.3 Scheduled 84
3.12.4 User libraries 85
3.12.5 Common functions 85
3.12.6 Start-up script 85
3.12.7 Tools 86
3.12.8 General scripting description 87
3.12.9 Script Editor 88
3.13 Alerts 90
3.14 Logs 91
3.15 Error Log 92
3.16 Help 92
4 Modbus RTU and Modbus TCP 93
4.1 Characteristics 93
4.3 Modbus Device Profiles 94
4.3.1 New profile definition 95

 2015 Schneider Electric. All rights reserved.

4.4 Modbus RTU Interface 97
4.4.1 Characteristics of Modbus RTU Interface 97
4.4.2 RS 485 Bus Topology Characteristics 97
4.4.3 Modbus RTU Configuration Commands 100
4.5 Modbus TCP Interface 103
4.5.1 Characteristics of Modbus TCP Interface 103
4.5.2 Modbus TCP configuration commands 103
4.6 Modbus Function Codes and Corresponding Master Functions 104
4.7 Modbus Master Functions 108
4.8 Modbus slave functions 109
4.8.1 Receive data from master 109
4.8.2 Set modbus mapping of slave device 109
4.8.3 Handle slave 109
4.8.4 Get functions 109
4.8.5 Set functions 109
4.8.6 Callback functions 109
5 Port Forwarding 111
5.1 Introduction 111
5.2 HTTP and HTTPS 111
5.2.1 HTTP 111
5.2.2 HTTPS 111
6 BACnet 112
6.1 Characteristics 112
6.2 homeLYnk Configuration 112
6.3 Object export 112
6.4 BACnet Configuration 113
6.4.1 Configuration 113
6.4.2 BACnet objects 113
6.5 BACnet Standardized Device Profile 114
6.5.1 List all BACnet Interoperability Building Blocks (BIBBs) Supported 114
6.5.2 BACnet Object Types Supported 114
6.5.3 Data Link Layer Options 114
6.6 Building Operation Workstation 115
7 RS-232 Serial Line 116

2015 Schneider Electric. All rights reserved.

7.1 Characteristics 116
7.2 Configuration Commands 116
8 USB 2.0 118
8.1 Characteristics 118
8.2 Configuration Commands 118
8.3 Send and receive SMS messages via attaching USB GSM adapter. 118
8.3.1 Command syntax: 119
9 Block Programming 120
9.1 Introduction 120
9.2 Block programming interface 120
9.3 Usage Examples 122
9.3.1 AND of two KNX objects 122
9.3.2 IF object value exceed maximum, alert is created. 122
9.4 Custom Function Blocks 123
9.5 Special Function Blocks 125
9.5.1 General 125
9.5.2 Convertors 125
9.5.3 Logical 125
9.5.4 Statistical 125
10 LUA – Programming Language 126
10.1 Object Functions 126
10.2 Group Communication Functions 127
10.3 Returned object functions 127
10.4 Data type Functions 128
10.5 Data Types 128
10.6 Data Storage Functions 129
10.7 Alert Functions 130
10.8 Log Functions 131
10.9 Time Functions 131
10.10 String Functions 131
10.11 Input and Output Functions 138
10.12 Script Control Function 138
10.13 Conversions 138
10.14 Bit Operators 139

 2015 Schneider Electric. All rights reserved.

10.15 Input and Output Facilities 139
10.16 Mathematical functions 140
10.17 Table manipulation 142
10.18 Operating System Facilities 143
10.19 Extended Function Library 144
11 Script Examples 147
11.1 Binary Filter 147
11.2 Binary Gate with Bit Gate 147
11.3 Gate with Byte Gate 148
11.4 Or - Port (2 in 1 0ut) 148
11.5 And - Port (2 in 1 0ut) 149
11.6 Or - Port (5 in 2 0ut) 149
11.7 And - Port (5 in 2 0ut) 150
11.8 Telegram Transformer (0/1 bit to 0-255 byte) 151
11.9 Compare Value 151
11.10 Save Scene 1 (RGB value) 151
11.11 Call Scene 1 (RGB value) 152
11.12 RGB object 152
11.12.1 How to operate the RGB object 152
11.12.2 Script for controlling RGB LED color 154
11.12.3 Script for reading status from the RGB LED 154
11.13 Hysteresis 155
11.14 Random byte value 155
11.15 Cyclic Repeater (delay 60 seconds) 156
11.16 Stepper / Counter Positive input 156
11.17 Stepper / Counter Negative input 156
11.18 Reset Stepper / Counter 157
11.19 On Delay (button set to 'update only internal') 157
11.20 Average 157
11.21 Off Delay 157
11.22 Stare Case Timer (with variable time object) 157
11.23 Value Memory (write to storage) 158
11.24 Value Memory (get from storage) 158
11.25 Multiplexer (1 in / 3 out) 158

 2015 Schneider Electric. All rights reserved.

11.26 Round Function Using Common Functions 158
11.27 Write Data and Time to KNX Group Addresses 159
11.28 Write Data to Groups with Tags 159
12 Application Note Documents 161
12.1.1 Actual List of Released Application Notes: 161
12.1.2 Application Notes Availability 161

2015 Schneider Electric. All rights reserved.

1 Product Security

1.1 Connectivity

- IP connection
- USB
- RS232
- RS485
- WIFI through IP connection and wireless router

1.2 Security Features

- Secure protocol HTTPS://IP:Port

- Network security must be set up at the appropriate level. homeLYnk should be part of a secure network with limited access. In case of connection to the Internet network is strictly recommended to use VPN or HTTPS channel.

- Security method is determined by the ability of other network elements (firewall, protection against virus and malware threats).

- It is strongly recommended to change password regularly e.g. every 90 days. The new password should be different and not similar to the previous one.

1.3 Operation

- It is strictly recommended to store the files containing your backups in a safe place without access of unauthorized persons.

- Entering the password should not be too simple, it should contain small / large letter, numeric character, at least 8 characters.

- In case you find a cyber security incidents or vulnerabilities, please contact us through this page:
http://www2.schneider-electric.com/sites/corporate/en/support/cybersecurity/contact-form.page

1.4 Maintenance

- In case of problems or questions regarding operation of homeLYnk, please contact your supplier or contact the Schneider Electric helpdesk in your country.
- Please be aware of higher security risk in case of remote access to your local network.

2015 Schneider Electric. All rights reserved.

1.5 Patch Management

- See chapter ***3.3.1*** for steps how to upgrade the firmware.

- Every upgrade must be manually performed. Please backup before an upgrade. See chapter ***3.2.7*** for backup procedure.

1.6 Factory Reset

- See chapter ***3.2.4*** for description how to reset the device.

1.7 Limitations

- Schneider Electric is not liable for damages caused by unqualified manipulation with the homeLYnk settings as well as not responsible for any security failure (e.g. by unauthorized access).

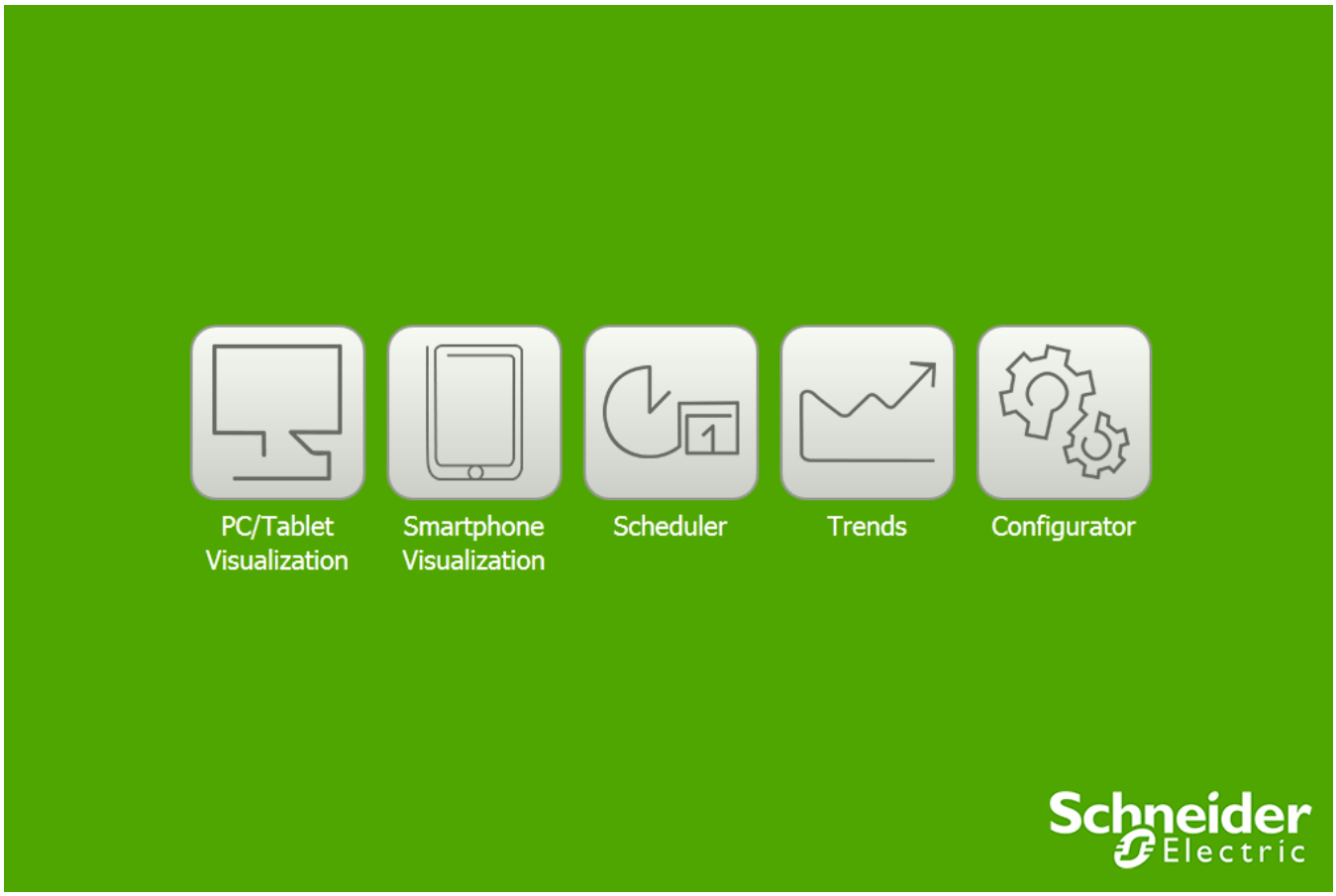
2015 Schneider Electric. All rights reserved.

2 Quick Start Guide

Follow the steps listed to help you get started with homeLYnk.

1. Mount the device on DIN rail.
2. Connect the bus cables (KNX, ModBus, and/or RS232) and/or flash drive.
3. Connect 24V power supply to the device (Positive conductor to the red clamp, negative conductor to the blue clamp).
4. Recommended accessory - Power supply REG/24V DC/0,4A, article No.: **MTN693003.**
5. Connect Ethernet cable from the PC.
6. Default IP address of the homeLYnk device is **192.168.0.10**. Change the IP address of the computer to the same range e.g. **192.168.0.9**; mask **255.255.255.0**.
7. Run Google Chrome or Mozilla Firefox (for OS Windows), Safari (for OS X) and go to **192.168.0.10.**

NOTE: Internet Explorer is not supported.



2015 Schneider Electric. All rights reserved.

2.1 Start Page

Start page is providing a dashboard-like view, pointing to the key areas of homeLYnk. The following options are located on the start page.

PC/Tablet Visualization – This icon navigates to the rich visualization with maps containing individual objects. It is ideal for PCs, iPads and Android tablets (preferably 10'' or bigger display size).

Smartphone Visualization – This icon navigates to the simple list visualization designed for iPhone/iPod/iPad/Android smart phones/ Android tablets (7'' or smaller display size). All objects which are added in homeLYnk ***Configurator*** by default are visible in this Smartphone visualization (if there is no ***Hide in Smartphone*** option enabled).

Scheduler – This icon navigates to a user friendly interface for the end-user to manage scheduler tasks for example, to specify thermostat values depending on the day of the week, time and holidays.

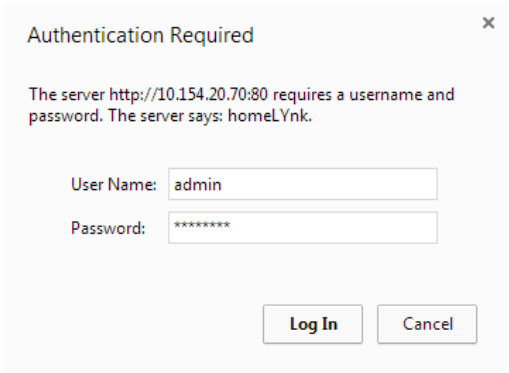
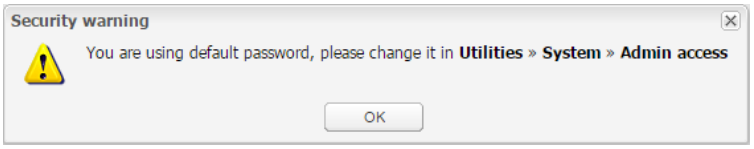
Trends – This icon navigates to a user friendly display of Trend logs with the ability to compare data between two different dates. It can display trends for up to 10 years.

Configurator – This icon navigates to programming, settings and configuration interface. Access is restricted to administrators.

NOTE: ***Configurator*** is not supported in mobile devices; it can be used only in desktop browsers.

2015 Schneider Electric. All rights reserved.

2.2 Default Configuration

<table>
<tr><td rowspan="2">homeLYnk Configurator
 
</td><td>Login</td><td>Password</td></tr>
<tr><td>admin</td><td>admin</td></tr>
<tr><td>NOTE: It is strongly recommended to change the default password. Reminder appears on every start of the homeLYnk Configurator with default password.</td><td colspan="2"> 
</td></tr>
<tr><td>IP address on LAN</td><td colspan="2">192.168.0.10</td></tr>
<tr><td>Networks mask on LAN</td><td colspan="2">255.255.255.0</td></tr>
</table>

2015 Schneider Electric. All rights reserved.

Change IP Settings

In ***Configurator*** → ***Utilities*** → ***System*** → ***Network*** → ***Interfaces*** window, click on the specific interface to change the IP settings.

Protocol – Specific protocol used for addressing:

- ***Static IP*** – Static IP address (default **192.168.0.10**).
- ***DHCP*** – DHCP protocol used to fetch IP configuration.

IP address – IP address received from the DHCP server. This field appears only if the IP address is given or else it is hidden.

Network mask – Network mask (default **255.255.255.0** (/24)).

Gateway IP – Gateway IP address.

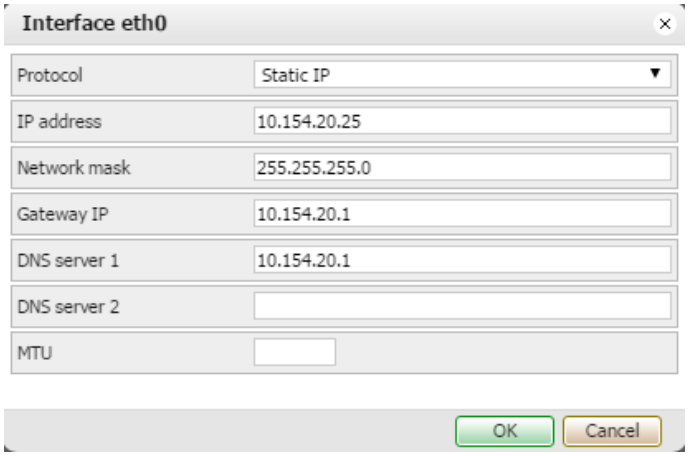
DNS server 1 – Primary DNS server IP address.

DNS server 2 – Secondary DNS server IP address.

Mtu – Maximum Transmission Unit, the largest size of the packet which is passed in the communication protocol (By default 150).

When changes are made, the

icon appears in the top-right corner. This should be applied for changes to take effect. homeLYnk will automatically reboot after these changes are applied.



2015 Schneider Electric. All rights reserved.

2.3 Discover homeLYnk IP Address

Windows PC

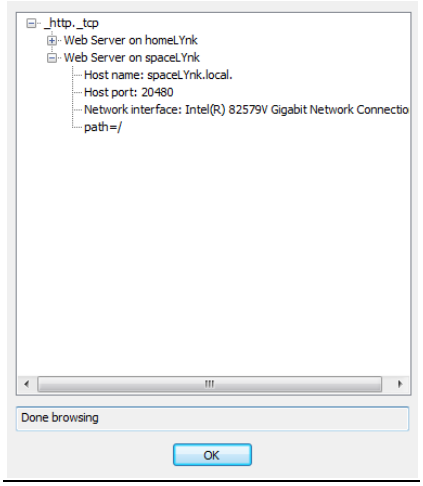
Option 1:

Use the utility Service Browser which can be downloaded here:

http://marknelson.us/attachments/2011/bonjour-windows/ServiceBrowserExe.zip

Apple bonjour is required (it is part of iTunes):

http://support.apple.com/kb/DL999



Option 2:

If host is not changed and only **one** homeLYnk is on the network, type in browser :

http://homeLYnk.local - Firefox

http://homeLYnk.local - Chrome

NOTE: Installed Bonjour service needed.

Linux PC

The utility called ***Avahi***, can be downloaded here: www.avahi.org

2015 Schneider Electric. All rights reserved.

Android The freely available app called ***ZeroConf Browser***, can be downloaded in Google Play.	
iOS/Mac OS The freely available app called ***Discovery bonjour browser*** can be downloaded from the *App Store*. For iPad install the iPhone/iPod version of the utility.	 http._tcp. Copy Web Server on spaceLYnk spaceLYnk.local:80 192.168.0.5 TXTRecord path = /

2015 Schneider Electric. All rights reserved.

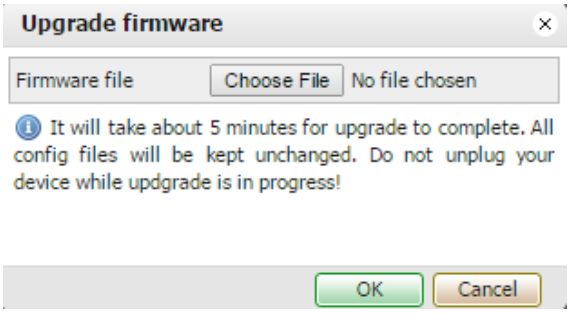
2.4 Firmware Upgrade

NOTE: Before each upgrade please backup the visualization, scripts and object in ***Configurator → Utilities → Backup***. During the upgrade the device will not respond as it will be rebooting.

After each upgrade, it is strongly recommended to clean the browser cache.

Use the web browser to perform the homeLYnk software upgrade. Firmware is available in the form of images and can be downloaded from the support page of SE office ***/Shopping kiosk***. (Available only for SE authorized persons).

Schneider recommends saving your project to an external drive after each project modification.

Complete System Upgrade ***Configurator → Utilities → System → System → Upgrade firmware***	

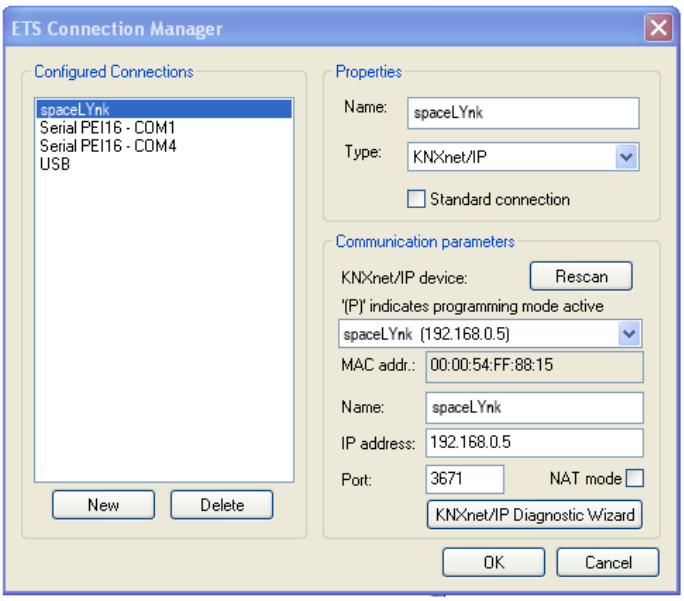
2015 Schneider Electric. All rights reserved.

2.5 homeLYnk for KNX/EIB Network Configuration Management with ETS3

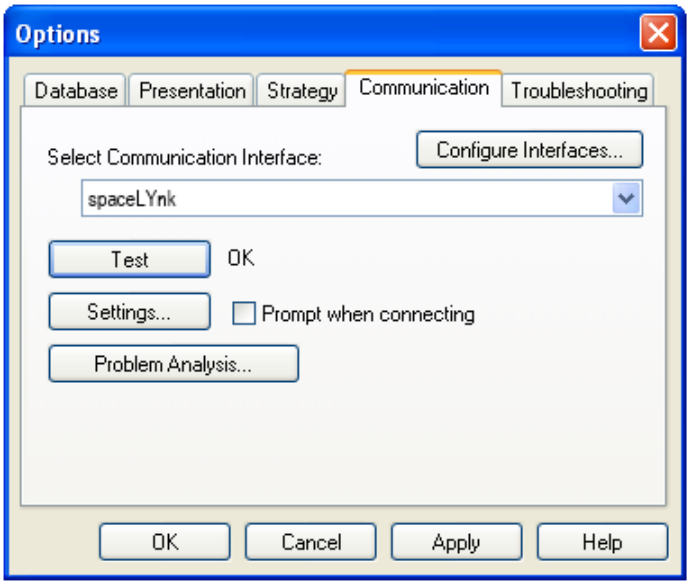
In order to use homeLYnk with KNXnet/IP functionality and to program with the other KNX bus devices, the device should be added into the ***ETS Connection Manager*.**

Extras → Options → Communication → Configure interfaces

1. Enter any *Name* for the connection.
2. Choose ***Type, and select KNXnet/IP from the drop-down.***
3. Press ***Rescan*** , and then select homeLYnk from the drop-down.
4. Press ***OK.***



5. In the ***Options → Communication*** window, select the newly created interface as ***Communication Interface*** from the drop-down.
6. To test the communication with ETS, press ***Test.***
7. Make sure that the bus status is Online – press button in ETS.



2015 Schneider Electric. All rights reserved.

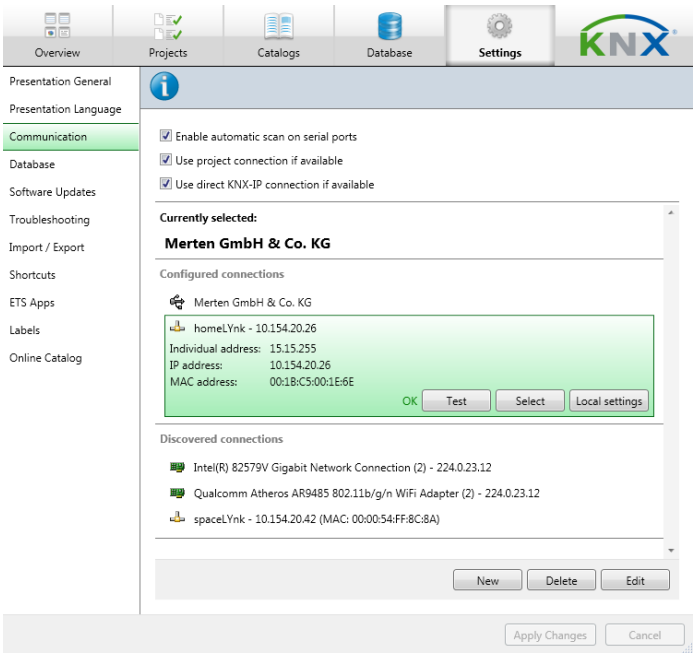
2.6 homeLYnk for KNX/EIB Network Configuration Management with ETS4

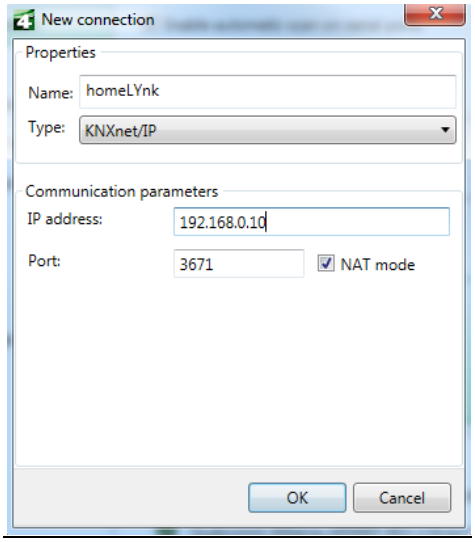
In order to use homeLYnk with KNXnet/IP functionality and to program with the other KNX bus devices, the device should be added into the ***ETS Connection Manager*****.** ETS programming through homeLYnk is available only when KNX IP features are enabled. Function “Bus monitor” is not included in homeLYnk.

Settings→ Communication

Newly added homeLYnk will be discovered automatically if it is connected in the same network as the PC running ETS4 software.

1. Choose ***Select*** to move it to the ***Configured connections***.
2. homeLYnk KNX individual address and mask can be set by pressing ***Local settings***.
3. Select ***Use project connection if available*** check box to make it a default project connection.
4. Select ***Use direct KNX-IP connection if available*** option for direct communication in IP network.
5. Press ***New*** to add homeLYnk manually.
6. Enter any ***Name*** for the device.
7. Set IP address, Port, and NAT mode (if needed).
8. Press ***OK*** to save changes.
9. To test the communication with ETS, press ***Test***.
10. Press ***Apply changes*** for changes to take effect.





2015 Schneider Electric. All rights reserved.

2.7 homeLYnk for KNX/EIB Network Configuration Management with ETS5

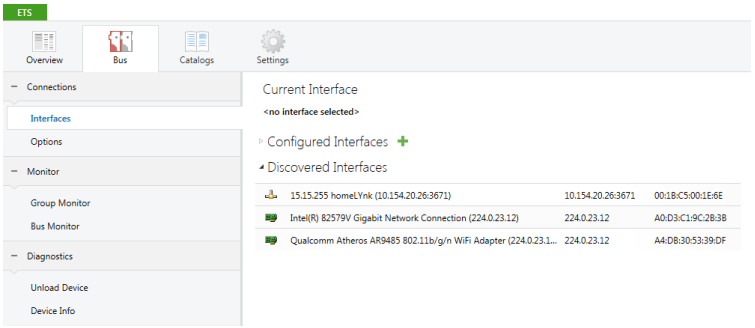
In order to use homeLYnk with KNXnet/IP functionality and to program with the other KNX bus devices, the device should be added into the ***ETS Connection Manager*****.** ETS programming through homeLYnk is available only when KNX IP features are enabled. Function “Bus monitor” is not included in homeLYnk.

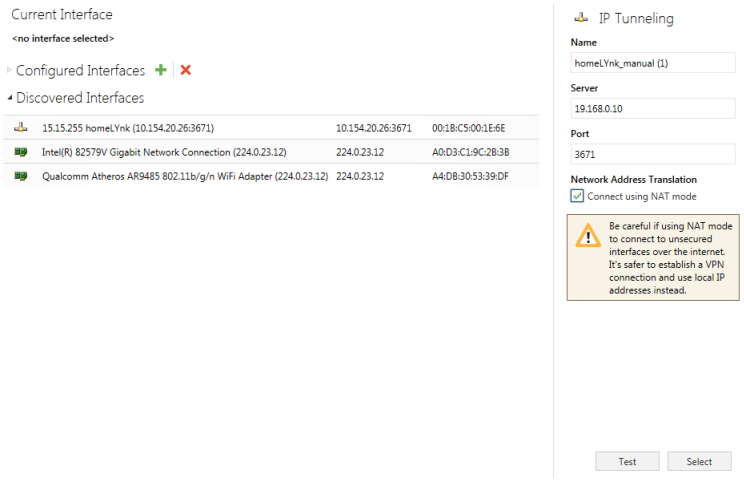
Bus → Connections → Interfaces

If your homeLYnk is in the same network with computer running ETS5, it is possible to discover the homeLYnk interface automatically. If your homeLYnk is discovered, choose the interface by double-click on item in ***Discovered Interfaces*** list.

If your interface is not discovered, follow steps below:

1. Click green + icon next to the ***Configured Interfaces.***
2. Select ***IP Tunneling***.
3. Click ***New Connection (0.0.0.0:3671***), which is created in ***Configured Interfaces***.
4. In the setting tab on right-hand side set ***Name*** of your connection, ***Server*** (IP address of homeLYnk) and ***Port***.
5. Select the interface, which you have configured in previous step.
6. To test the communication with ETS, press ***Test*** in lower-right corner.
7. If test is OK, select the interface as active by double click on item in list of ***Configured Interfaces***.
8. ***Current Interface*** is set.




Current Interface
<no interface selected>
Configured Interfaces
Discovered Interfaces
15.15.255 homeLYnk (10.154.20.26:3671) 10.154.20.26:3671 00:1B:C5:00:1E:6E
Intel(R) 82579V Gigabit Network Connection (224.0.23.12) 224.0.23.12 A0:D3:C1:9C:2B:3B
Qualcomm Atheros AR9485 802.11b/g/n WiFi Adapter (224.0.23.12) 224.0.23.12 A4:DB:30:53:39:DF
IP Tunneling
Name
homeLYnk_manual (1)
Server
19.168.0.10
Port
3671
Network Address Translation
Connect using NAT mode
Be careful if using NAT mode to connect to unsecured interfaces over the internet. It's safer to establish a VPN connection and use local IP addresses instead.
Test
Select

2015 Schneider Electric. All rights reserved.

2.8 KNX and IP Router Settings

KNX specific configuration is located in:

Configurator → Utilities → System → Network → KNX connection.

General Tab

Mode: KNX connection mode.

- ***TP-UART-*** (built-in by default) – Physical connection via twisted pair.
- ***EIBnet/IP Tunneling-*** Point-to-point connection over a communication channel is established between the EIBnet/IP server and client.
- ***EIBnet/IP Tunneling (NAT mode) –*** Tunnelling mode with Network address translation.
- ***EIBnet/IP Routing-*** Exchange of KNX/EIB telegrams between different EIBnet/IP servers. An EIBnet/IP server forwards only those telegrams that correspond to the filter criteria. Routing uses multicast telegrams.

ACK all group telegrams: If selected, then homeLYnk acknowledges every group telegram.

KNX address: KNX physical address of the device.

KNX IP features: Use this device with KNX IP features. For example, KNXnet/IP network configuration.

Multicast IP: Multicast IP address.

Multicast TTL: Multicast TTL (Time To Live).

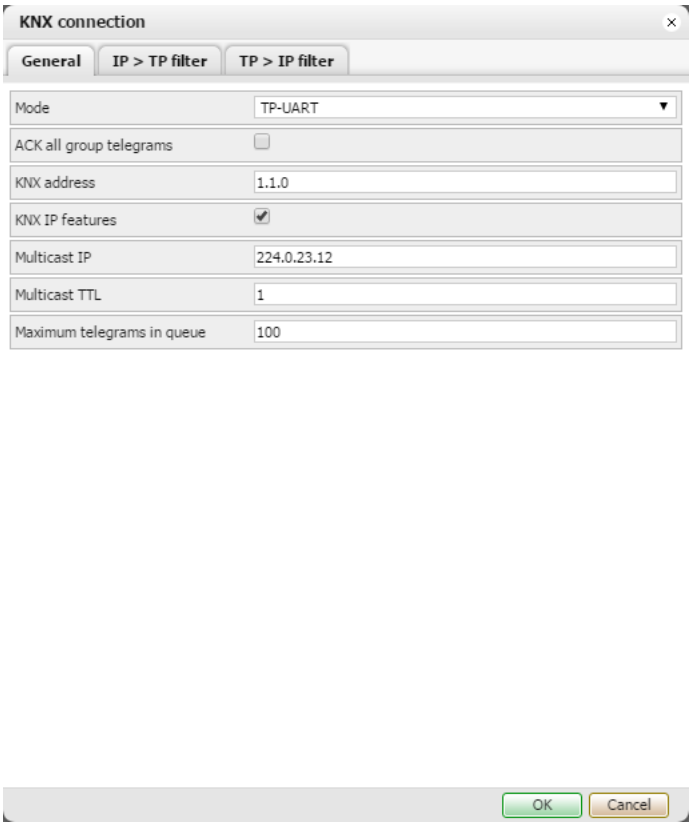
Maximum telegrams in queue: Count of maximum telegrams in the queue.

NOTE: If KNX TP is not connected to the device, ***Routing*** mode should be used to update the group addresses correctly.

When changes are done, the

icon appears in the top-right corner.

This should be applied for changes to take effect. homeLYnk will automatically reboot after these changes are applied.



2015 Schneider Electric. All rights reserved.

2.9 Create Visualization for iPad/PC

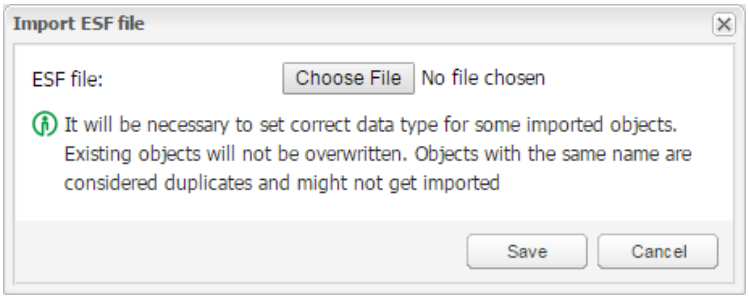
2.9.1 Import Objects

In order to complete ETS3 project, generate *.ESF file from ETS through:

File → Extract data →Export to OPC server

Import *.ESF file to

Configurator → Utilities → Import ESF file


Import ESF file
ESF file:
Choose File
No file chosen
It will be necessary to set correct data type for some imported objects. Existing objects will not be overwritten. Objects with the same name are considered duplicates and might not get imported
Save
Cancel

In order to complete ETS4 or ETS5 project, generate* .ESF file from ETS project through:

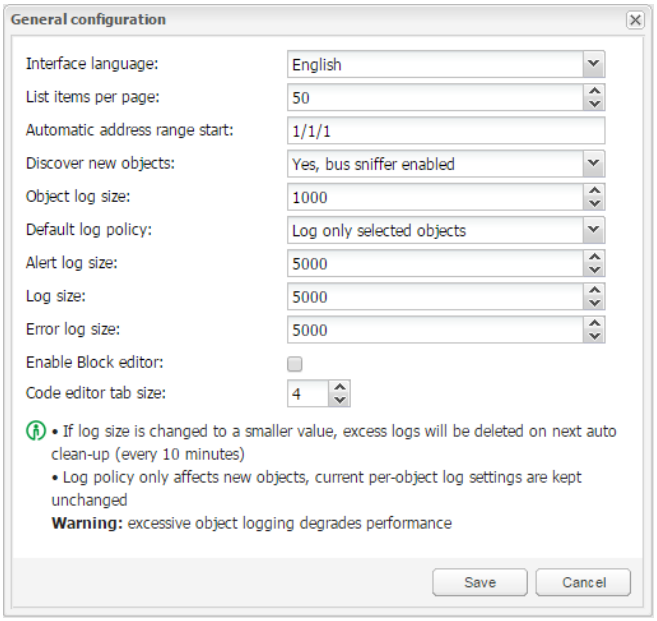
Extras →Export OPC

Import *.ESF file to:

Configurator → Utilities → Import ESF file

Or, connect homeLYnk to the KNX bus and it will detect objects automatically in the ***Objects*** tab once they are activated (KNX telegram is sent).
This option can be Enabled/Disabled in ***Configurator → Utilities → General Configuration → Discover new objects***

NOTE: Objects can be added manually as well.


General configuration
Interface language: English
List items per page: 50
Automatic address range start: 1/1/1
Discover new objects: Yes, bus sniffer enabled
Object log size: 1000
Default log policy: Log only selected objects
Alert log size: 5000
Log size: 5000
Error log size: 5000
Enable Block editor:
Code editor tab size: 4
• If log size is changed to a smaller value, excess logs will be deleted on next auto clean-up (every 10 minutes)
• Log policy only affects new objects, current per-object log settings are kept unchanged
Warning: excessive object logging degrades performance
Save
Cancel

2015 Schneider Electric. All rights reserved.

2.9.2 Create 'Building/Floor' Structure and Add Objects to the Map

Go to ***Configurator*** → ***Vis.structure***

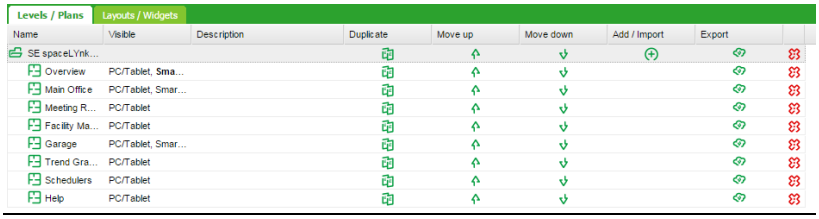
Vis. Structure allows creating all building levels and visualizations plans. Additionally, it can create Layouts and Widgets for visualization plans.

Starting new project, only ***Layouts*** and ***Widgets*** folders are visible. Adding new level allows, defining a specific ***Plan*** of 'flat'. ***Layouts*** and ***Widgets*** are additional tools which are not mandatory for basic visualization; they can be defined and implemented in many other ***Plans***.

Levels

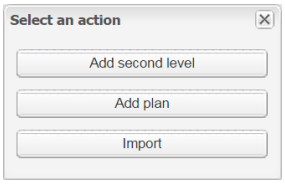
To add new Level press

button**.** Main level usually is the project name (additional levels can be added later).



Second level

If additional levels are needed, press on next to the main level.



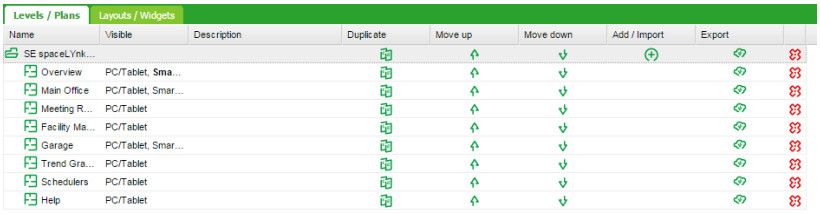
Select **Add second level** and provide name and sort order.

Each level can be duplicated with sublevels and plans by pressing the duplicate icon next to the level.

2015 Schneider Electric. All rights reserved.

Plans

To add Plans press on , and select ***Add plan***.



Parent: Parent directory.

Name: Name of the plan.

Plan size: Size of the plan. User defined or predefined from list.

Layout: Layout for the plan. All ***Objects*** from the Layout will be duplicated on the plan including the background color and the plan image, if they are not defined separately.

PC/Tablet visualization [Show, Show, make default, Hide]: Visibility for this particular plan in PC/Tablet visualization.

Smartphone visualization [Show, Show, make default, Hide]: Visibility for this particular plan in Smartphone devices visualization.

Pin code: Set the code, which is needed to enter this particular plan of visualization. If it is not defined, no code is asked by entering the visualization plan.

Primary background image: Select the background previously added to ***Vis. Graphics → Images/Backgrounds.***

Secondary background image: Select the secondary background previously added to ***Vis. Graphics → Images/Backgrounds*** for parallax projection.

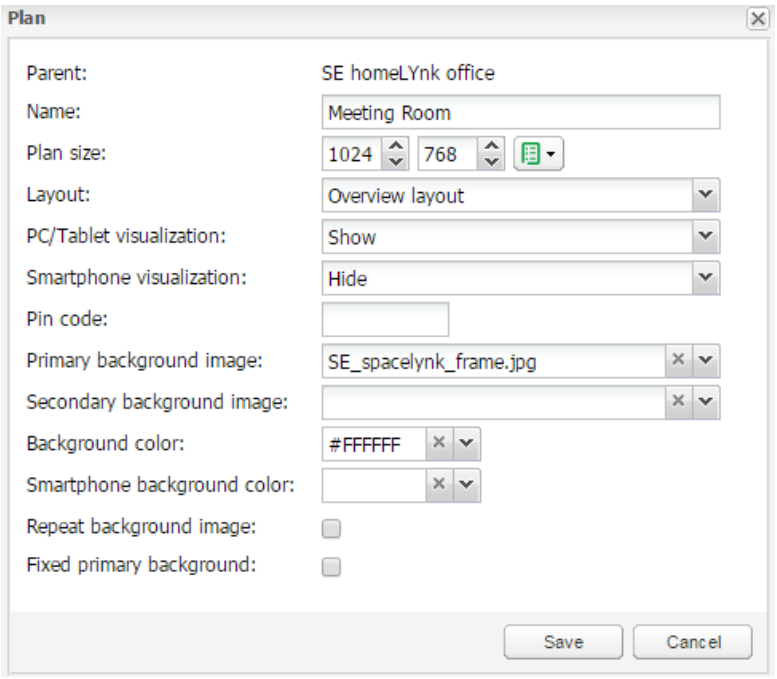
Background color: Choose the background color of the plan for PC/Tablet Visualization.

Smartphone background color: Choose the background color of the plan for Smartphone Visualization.

Repeat background image: To show the image once or repeat it and fill the whole plan.

Fixed primary background: Primary plan is static in parallax projection.

NOTE: Each plan can be duplicated together with all components on a plan by pressing duplicate icon next to the plan .

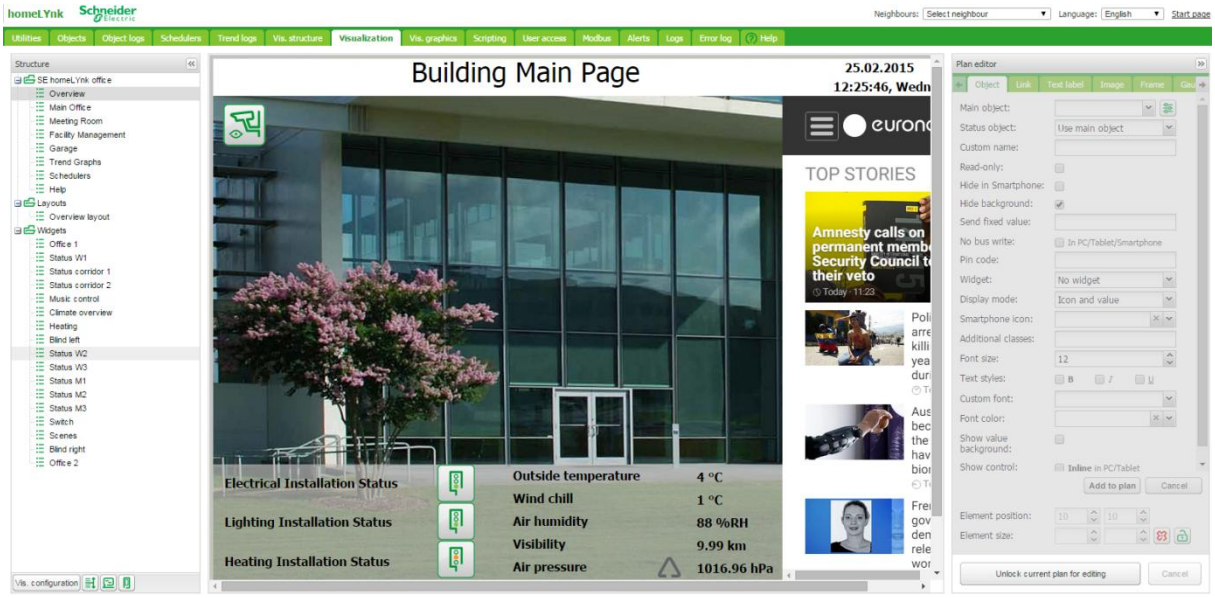


2015 Schneider Electric. All rights reserved.

2.9.3 Add Objects to Newly Created Visualization Map

Go to ***Configurator → Visualization***.

Once the ***Level*** and plan's structure are defined in ***Vis.structure*** tab, it can be visualized in the ***Visualization*** tab. Controlled and monitored objects can be added and managed in this section. Both sidebars can be minimized by pressing the left or the right arrow icon which will make the map appear more visible especially on smaller displays.



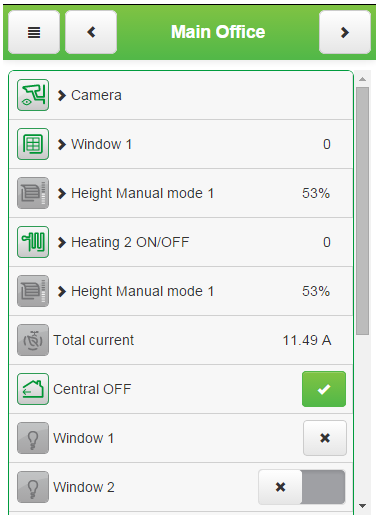
Existing objects can be added to the map by clicking on ***Unlock current plan for editing***. After defining the object parameters, press ***Add to plan*** and a newly created object appears. This object can be moved to the desired location but whilst in editing mode the object will not work. When all the necessary objects are added, press ***Save and reload plan***, so that the objects can be visualized.

2015 Schneider Electric. All rights reserved.

2.9.4 Launching Visualization on Smartphone Device (iPod in this case)

In order to do so, please follow the next steps:

1. Make sure the iPod is connected wirelessly to the homeLYnk (through separate access point – wireless router).
2. Enter homeLYnk IP (default **192.168.0.10**) in iPod web browser.
3. Click on the **Smartphone visualization** icon .
4. Application's link can be saved. Provide a shortcut in the iPod for easy access.



2.9.5 Launching vis. on PC, Tablet or Any Other Touch Device with Large Screen

In order to do so, please follow the next steps:

1. Ensure the PC/Tablet device is able to access homeLYnk, and enter the IP in the browser (default **192.168.0.10**).
2. Click on ***PC/Tablet visualization***
3. Select the desired ***Plan***.
4. Sidebar can be minimized by pressing on icon to make the map more visible.

2015 Schneider Electric. All rights reserved.

3 Advanced Guide

3.1 Configurator's Main Page

Configurator's main page - top bar:

Neighbours - Switch to next homeLYnk in the same network. This selection appears only if any other homeLYnks or homeLYnks are discovered.

Language - Switch language of the operational interface to English, Czech, Danish, Dutch, French, German, Italian, Portuguese, Russian, Spanish or Turkish.

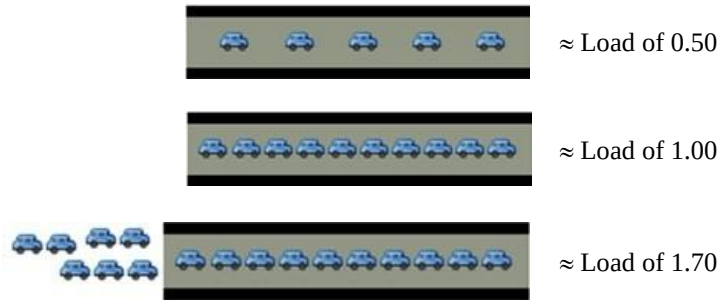
Start page - Link to the main visualization page.

Configurator's main page - bottom bar:

Version: 1.3 - This is the actual firmware version of the homeLYnk.

CPU/IO: 0.59 0.44 0.2|5, Memory 11% - Load average numbers ***0.59 0.44 0.25*** represent averages over progressively longer periods of time (one, five and fifteen minute average). The lower number the better.

Bridge traffic analogy to processes:



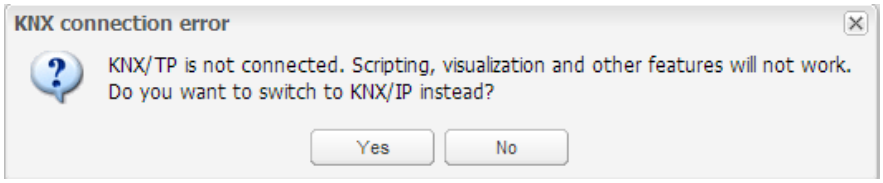
NOTE: Inspect your running tasks if the load exceeds the level **0.70**!

2015 Schneider Electric. All rights reserved.

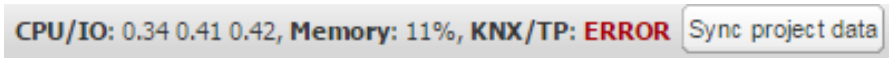
Memory $= \frac{Used\ memory - Buffered - Cached}{Total\ system\ memory}$ (minimum occupied memory in %)

See ***System / Status / System status / Memory usage***. Beware of Linux terminology. Linux calls *cached* and *buffered* memory "*used*" even if it could be understood as "*free*" for new applications.

KNX/IP: Each time the Configurator is opened, the homeLYnk checks if the KNX bus is connected. If not connected, then an error message appears stating that: Scripting, visualization and other features will not work. Do you want to switch to KNX/IP instead?


KNX connection error
KNX/TP is not connected. Scripting, visualization and other features will not work.
Do you want to switch to KNX/IP instead?
Yes
No

Selected connection and its status are visible in the right bottom corner:



NOTE: KNX interface has to be changed back to TP-UART once KNX bus is connected under ***System →Network →KNX connection →Mode***. KNX interface change must be confirmed by rebooting homeLYnk (manually or pressing the

button).

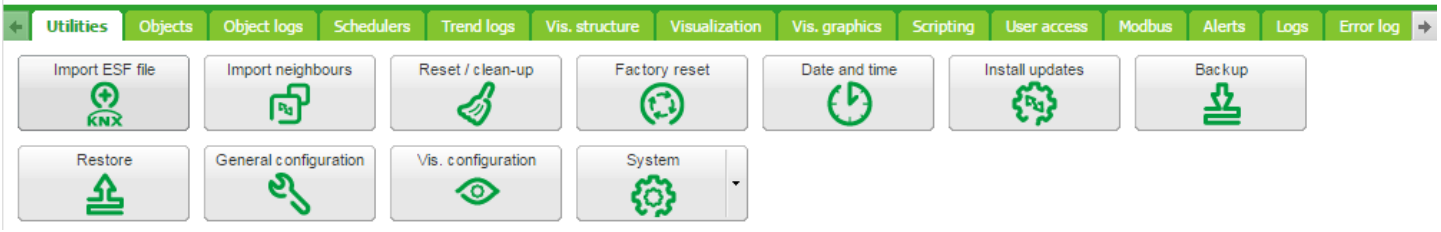
Sync project data: This button is useful after some bigger change in the project. When pressed, project will be immediately synchronized to the microSD card.

Automatic synchronization is performed every 30 minutes only and all unsaved changes may be lost.

2015 Schneider Electric. All rights reserved.

3.2 Utilities

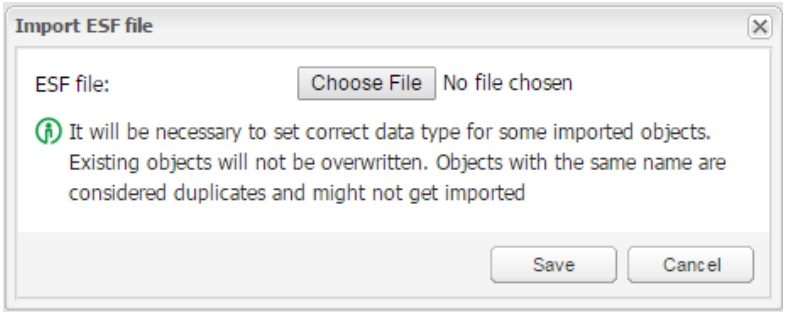
Below is the Utilities tab screenshot.

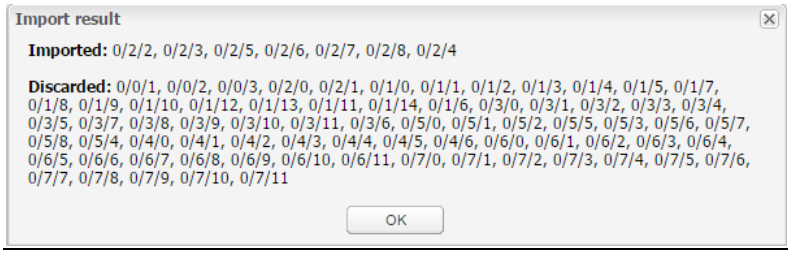


3.2.1 Import ESF file

Imports the ETS object file. It is essential to set correct data types for imported objects. Existing objects will not be overwritten. Objects with the same name are considered duplicates and might not be imported. List of imported and discarded object is displayed after each import.

(See chapter 1.8.1 for ETS3, ETS4 and ETS5 import details.)

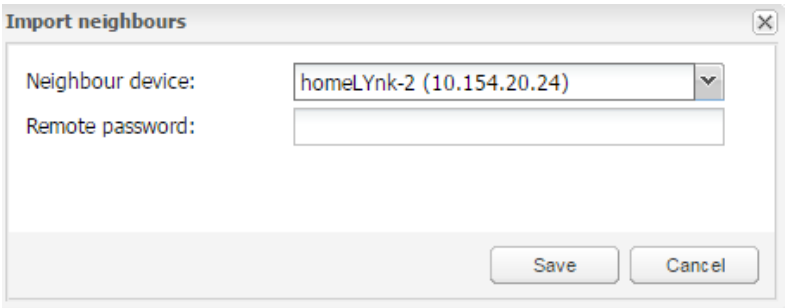

Import ESF file
ESF file:
Choose File
No file chosen
It will be necessary to set correct data type for some imported objects. Existing objects will not be overwritten. Objects with the same name are considered duplicates and might not get imported
Save
Cancel



3.2.2 Import Neighbours

If there is more than one homeLYnk on the same IP network, ***Import neighbours*** button will appear under ***Utilities***. It allows importing the exported objects from another homeLYnk. System will ask for ***Remote password*** of the second device from which data will be exported.

Export option must be activated for ***Objects*** to be shared between the homeLYnks. Enabling will make those objects visible via BACNet and remote services (XML/JSON).

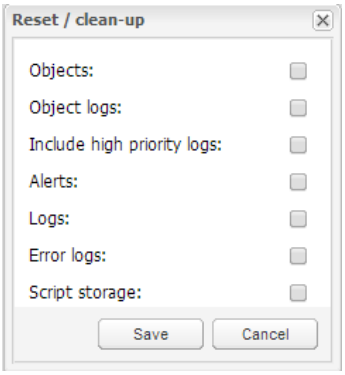


2015 Schneider Electric. All rights reserved.

3.2.3 Reset / clean-up

Delete selected items from the homeLYnk. If you select ***Objects***, they will be deleted from the visualization part as well.

NOTE: Backup all important data before proceeding with Reset /clean-up.



3.2.4 Factory reset

Delete all configurations and reset to the factory default settings. System settings will stay unchanged.

To perform **total** factory reset, double long press on the RESET button located on the back side of the homeLYnk device (10 seconds press, release and press again for 10 seconds).

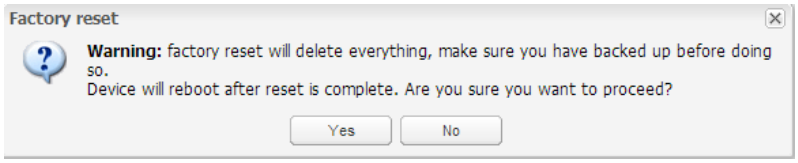
IP address after HW factory reset with HW button is always 192.168.0.10.

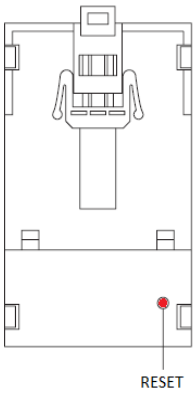
When factory reset is performed by SW button in the main menu

IP address will stay unchanged and visualization graphic will be kept as well.

For default IP address with preserved project press HW reset button for 10s.

NOTE: Total reset with hardware button is meant especially for situations where homeLYnk is not accessible due to wrong settings.


Factory reset
Warning: factory reset will delete everything, make sure you have backed up before doing so.
Device will reboot after reset is complete. Are you sure you want to proceed?
Yes
No



2015 Schneider Electric. All rights reserved.

3.2.5 Date and time

Network time protocol (NTP) is implemented. Along with the internet connection, homeLYnk will automatically update time from servers defined in ***Utilities/System/Services/NTP client***:

0.europe.pool.ntp.org

1.europe.pool.ntp.org

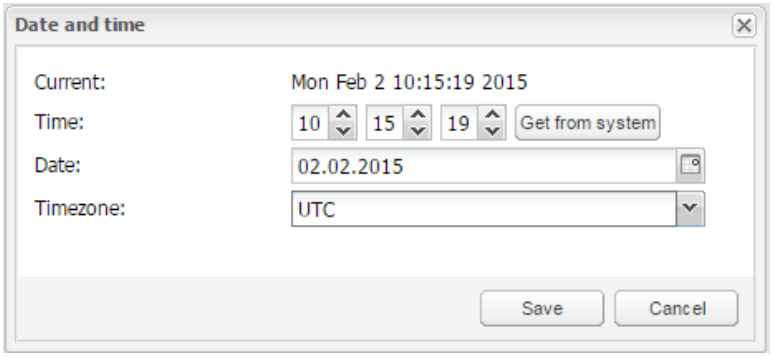
2.europe.pool.ntp.org

3.europe.pool.ntp.org

NOTE: It is important to select the correct time zone.

If there is no internet connection, click on ***Get from the system*** to adopt time from the PC.

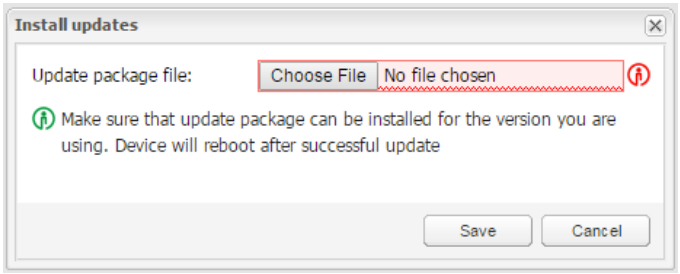
NOTE: Without power, homeLYnk will maintain time and date settings only for a limited period of time (app. 5 minutes).



3.2.6 Install updates

For partial updates or adding of pre-fabricated solutions.

Install homeLYnk update file *.lmup. homeLYnk will reboot after the successful update.


Install updates
Update package file:
Choose File
No file chosen
Make sure that update package can be installed for the version you are using. Device will reboot after successful update
Save
Cancel

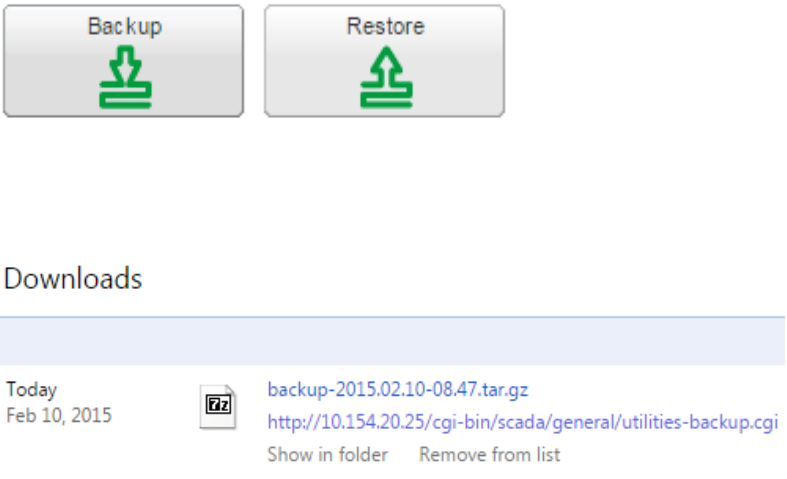
3.2.7 Backup

Backup all the objects, trends, logs, scripts, icons, images, backgrounds, visualization and KNX filter table to the backup-dd.mm.yyyy-hh.mm.tar.gz file (actual homeLYnk time and date is used when the backup is generated).

Created backup file is placed into the browser **Downloads** folder.

File can be renamed to match project structure.

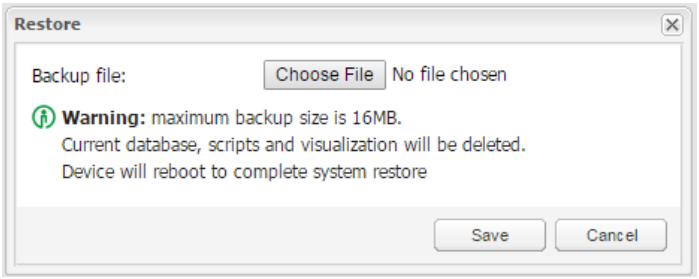
NOTE: Maximum backup size is 16MB. Check the size of the backup once you create it. If it is bigger than 16MB, you won't be able to restore it. System configuration, network settings, passwords or KNX settings are not backed up. Filter table is backed up.



2015 Schneider Electric. All rights reserved.

3.2.8 Restore

Restores configuration from backup.



3.2.9 General Configuration

Interface language: Interface language (English, Czech, Danish, Dutch, French, German, Italian, Portuguese, Russian, Spanish and Turkish).

List items per page: Count of lines per page for example, Objects, Object logs, Alerts etc. (Maximum value is 50).

Automatic address range start: Newly added group objects will start addressing from defined range.

Discover new objects: KNX object sniffer is enabled. If YES is selected, all new objects automatically appear in the Objects list.

Object log size: Count of object logs. (Maximum value is 10000).

Default log policy: Log status for all objects or only for checked objects can be selected.

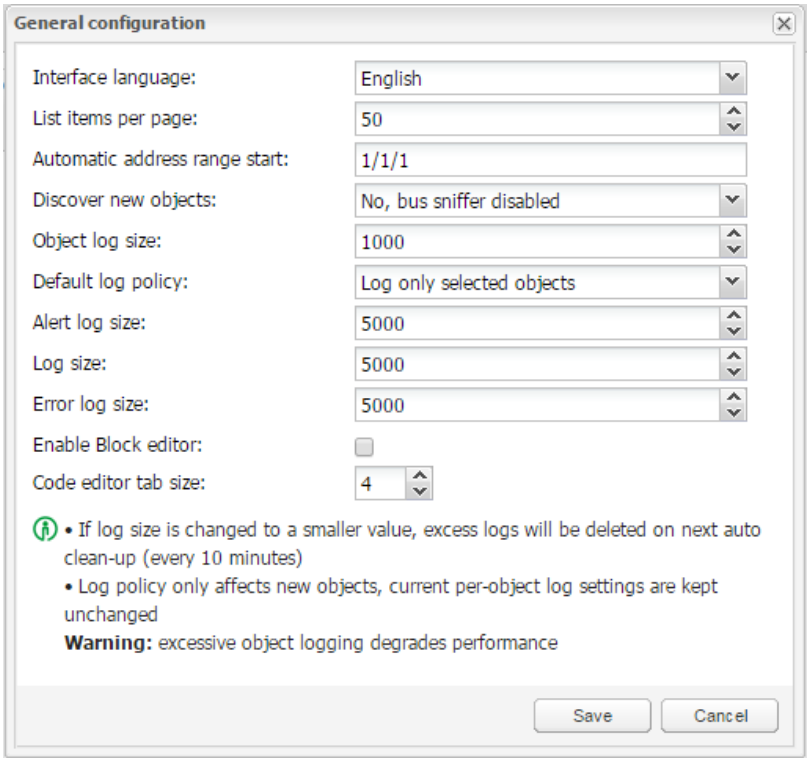
Alert log size: Count of alerts logged. (Maximum value is 5000).

Log size: Count of logs. (Maximum value is 5000).

Error log size: Count of errors logged. (Maximum value is 5000).

Enable block editor: Enabling/disabling function block editor in Scripting.

Code editor tab size: Pressing TAB in scripting editor results in insertion of defined count of spaces



NOTE:

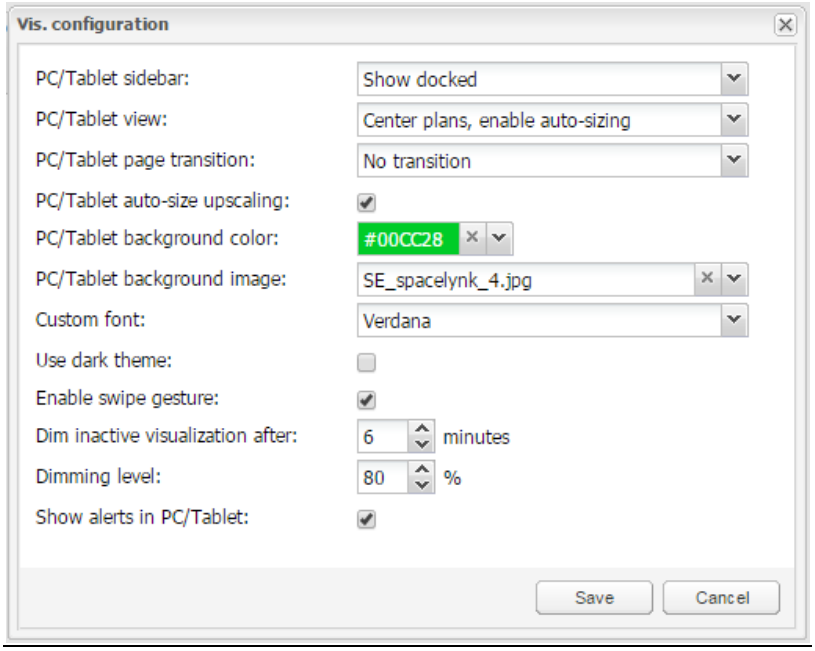
homeLYnk reboot is required when changing ***List items per page*** or ***Language*** parameter.
(Clear the browser cache).

homeLYnk keeps the log objects above the limit for 15 minutes; after the time elapse, all records above the limit will be cleared. It is necessary to be aware while logging large amount of data in time.

Excessive object logging degrades homeLYnk performance.)

2015 Schneider Electric. All rights reserved.

3.2.10 Vis.configuration



PC/Tablet sidebar: **[Show docked / Show as overlay / Hide]** Enable sidebar with list of plans in visualization docked/with auto-hide option/ hidden.

PC/Tablet view: **[Align plans to top left, no size limits / Center plans, limit size /Center plans, enable auto-sizing / Center horizontally, auto size width]**

NOTE: Auto sizing works only in web browsers with Web Kit engine (Chrome, Safari) and Firefox.

PC/Tablet page transition: **[No transition / Flip X / Flip Y / Shrink / Expand / Slide Up / Slide Down / Slide Left / Slide Right / Slide Up Big / Slide Down Big / Slide Left Big / Slide Right Big]** Selection of transition effect for page changing in visualization.

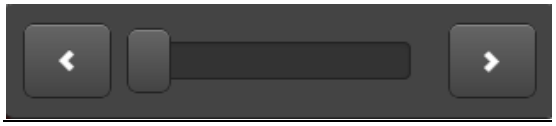
PC/Tablet auto/size upscaling: Enable automatic re/scaling for multiple screen resolutions.

PC/Tablet background color: Common background color for visualization.

PC/Tablet background image: Common background image for visualization.

Custom font: Common font for visualization.

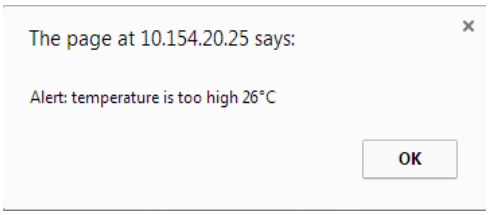
Use dark theme: Inversion of colors, fonts and controls to match dark palettes visualizations.



Dim inactive visualization after:? minutes: Feature to save energy of battery powered devices.

Dim inactive visualization after:? minutes: Feature to save energy of battery powered devices.

Dimming level ?%: Brightness level of dimmed screen.

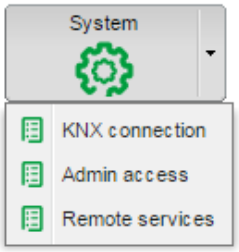
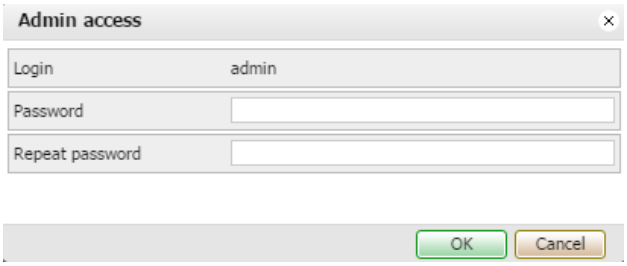
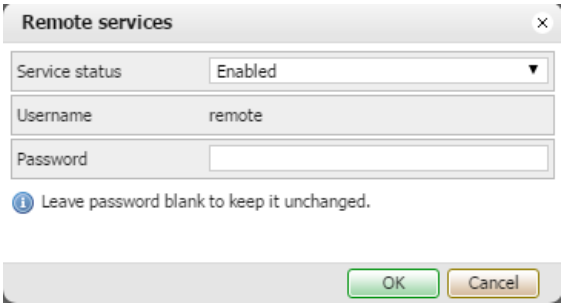


***Show alerts in PC/Tablet*:** After triggering new alerts, it will appear in PC/Tablet visualization.

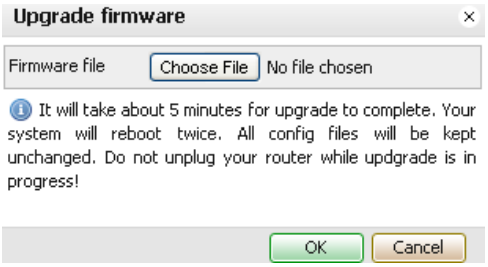
2015 Schneider Electric. All rights reserved.

3.3 System

3.3.1 System → System

System Allows managing router functionality on KNX/EIB homeLYnk as well as access control management and firmware upgrade.	
KNX connection See chapter *3.3.2* for details.	
Admin access Password settings for administrator account. Username is admin by default. The login and password configuration for ***User access*** is located in main menu.	
Remote services For enabling/disabling remote access of homeLYnk for service purposes.	

2015 Schneider Electric. All rights reserved.

Upgrade firmware ***System → Upgrade firmware*** is used to perform complete upgrade of the system (both OS as well as homeLYnk part). **NOTE:** After each upgrade, it is strongly recommended to clean the browser cache. During firmware upgrade the device will not respond, because homeLYnk will reboot several times.	
Reboot By executing ***System → Reboot*** command, homeLYnk will restart.	
Shutdown By executing ***System → Shutdown*** command, homeLYnk will shut down. **NOTE:** It is strongly advised to shutdown the system before the unit is powered off, so that the database can be saved securely. The system is shutdown, when LED no. 1 stops blinking. The only way to switch homeLYnk ON again is to disconnect and re-connect power supply. It can't be switched ON remotely.	

2015 Schneider Electric. All rights reserved.

3.3.2 System → Network

Interfaces

Ethernet interface is listed in the first tab. Traffic flowchart can be opened by using graph button on the right side.

By clicking on the interface, the configuration window appears.

Protocol – Specific protocol used for addressing.

- ***None*** – No protocol is used.
- ***Static IP*** – Static IP address. By default **192.168.0.10**.
- ***DHCP*** – Use DHCP protocol to get IP configuration.
- ***Current IP*** – The IP address got from DHCP server. This field appears only if the IP address is given otherwise it is hidden.

IP address – By default **192.168.0.10**.

Network mask – Network mask. By default is **255.255.255.0** (/24).

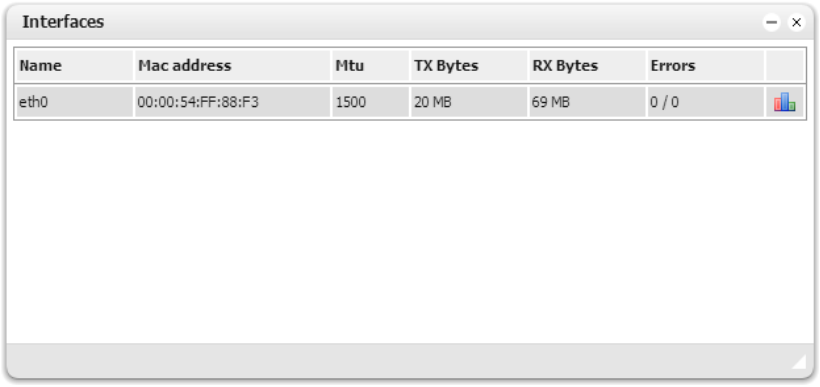
Gateway IP – Gateway IP address.

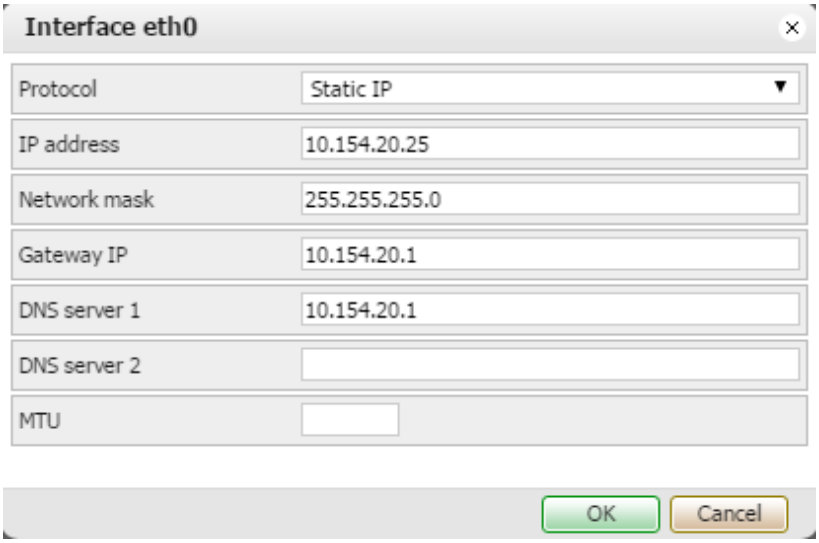
DNS server 1 – Primary DNS server IP address.

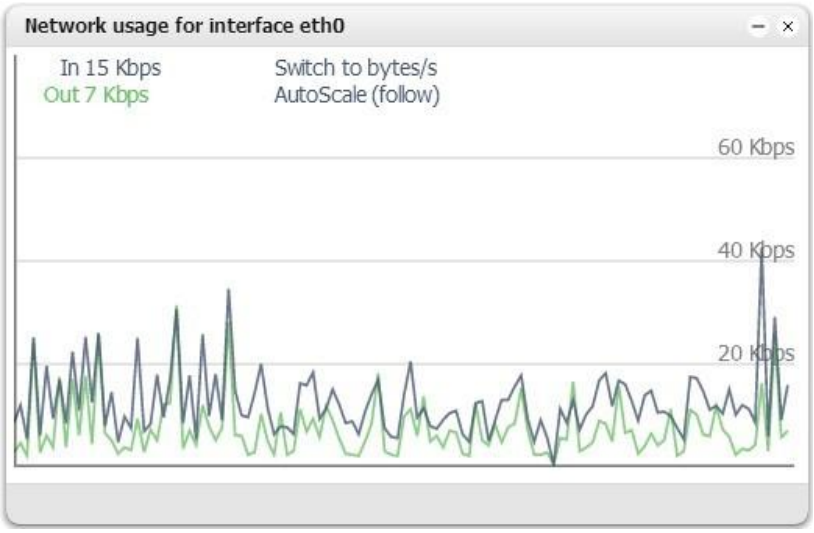
DNS server 2 – Secondary DNS server IP address.

Mtu – Maximum transmission unit, the largest size of the packet which could be passed in the communication protocol. (Default 1500).

Ethernet interface data put through graph - On the main window of the **Ethernet** tab, if you click on the button, a new window is opened. It draws a real-time graph of the traffic flow passing the interface (both In and Out). There is a possibility to switch the units of measurement – bite/s or Byte/s and graph Auto Scale follow or Up.



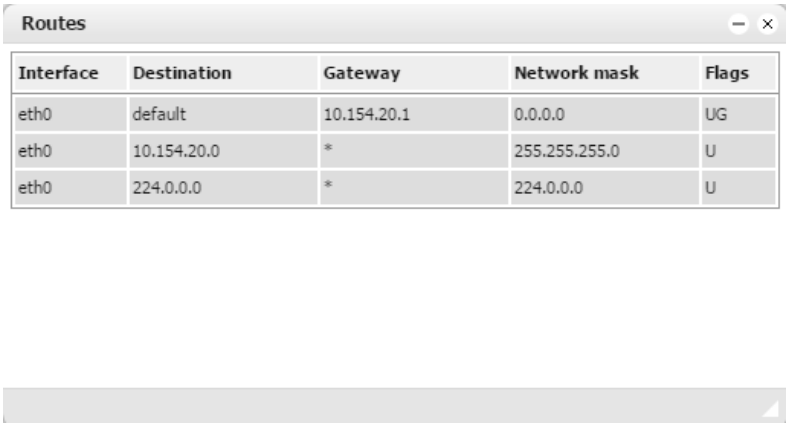




2015 Schneider Electric. All rights reserved.

Routes

Routing table is a data table that lists the routes to a particular network destination. It contains information about the topology immediately around it. System routing table is located in **Network→ Routes menu**. The window is divided in two parts –**Dynamic** and **Static** routes.



- ***Dynamic***

List of self learned network destinations and automatic selection of the 'best route'.

Interface – Interface name indicates the locally available interface that is responsible for reaching the gateway.

Destination – Destination subnet IP address describes together with Network mask the Network ID.

Gateway – Gateway IP address points to the gateway through which the network can be reached.

Network mask – Network mask.

Flags –Helps in troubleshooting your network problem, see the attached coding table.

- ***Static***

Manual entering of routes into the homeLYnk routing table, they do not change automatically.

Interface – Interface name.

Destination – Destination IP address.

Gateway – Gateway IP address.

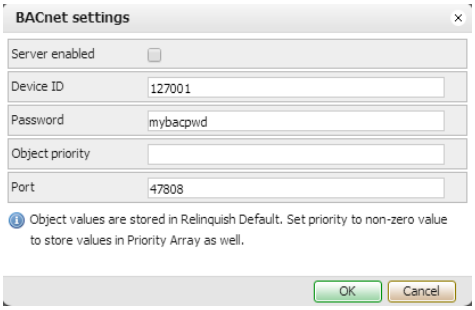
Network mask – Network mask.

Flags – Helps in troubleshooting your network problem, see the attached coding table.

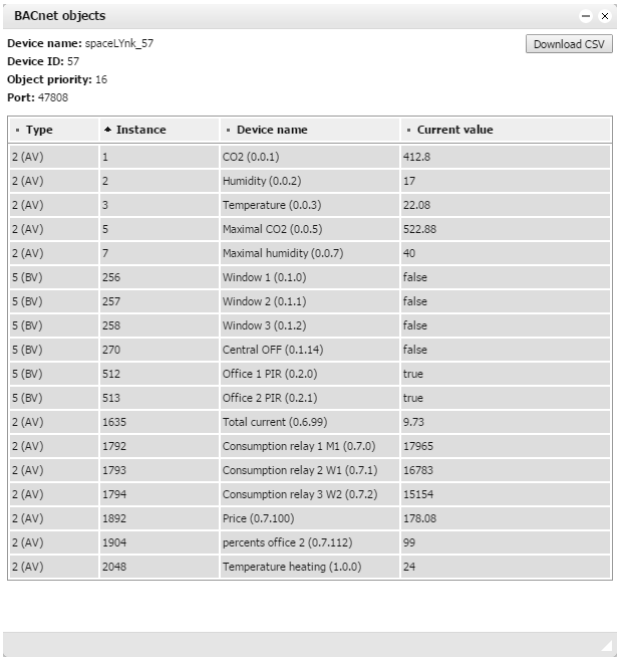
Flag	Name	Meaning
1	RTF_PROTO1	Protocol specific routing flag 1
2	RTF_PROTO2	Protocol specific routing flag 2
3	RTF_PROTO3	Protocol specific routing flag 3
B	RTF_BLACKHOLE	Just discard pkts (during updates)
b	RTF_BROADCAST	The route represents a broadcast address
C	RTF_CLONING	Generate new routes on use
c	RTF_PRCLONING	Protocol-specified generate new routes on use
D	RTF_DYNAMIC	Created dynamically (by redirect)
G	RTF_GATEWAY	Destination requires forwarding by intermediary
H	RTF_HOST	Host entry (net otherwise)
L	RTF_LLINFO	Valid protocol to link address translation
M	RTF_MODIFIED	Modified dynamically (by redirect)
R	RTF_REJECT	Host or net unreachable
S	RTF_STATIC	Manually added
U	RTF_UP	Route usable
W	RTF_WASCLONED	Route was generated as a result of cloning
X	RTF_XRESOLVE	External daemon translates proto to link address

2015 Schneider Electric. All rights reserved.

BACnet settings



BACnet objects



Type	Instance	Device name	Current value
2 (AV)	1	CO2 (0.0.1)	412.8
2 (AV)	2	Humidity (0.0.2)	17
2 (AV)	3	Temperature (0.0.3)	22.08
2 (AV)	5	Maximal CO2 (0.0.5)	522.88
2 (AV)	7	Maximal humidity (0.0.7)	40
5 (BV)	256	Window 1 (0.1.0)	false
5 (BV)	257	Window 2 (0.1.1)	false
5 (BV)	258	Window 3 (0.1.2)	false
5 (BV)	270	Central OFF (0.1.14)	false
5 (BV)	512	Office 1 PIR (0.2.0)	true
5 (BV)	513	Office 2 PIR (0.2.1)	true
2 (AV)	1635	Total current (0.6.99)	9.73
2 (AV)	1792	Consumption relay 1 M1 (0.7.0)	17965
2 (AV)	1793	Consumption relay 2 W1 (0.7.1)	16783
2 (AV)	1794	Consumption relay 3 W2 (0.7.2)	15154
2 (AV)	1892	Price (0.7.100)	178.08
2 (AV)	1904	percents office 2 (0.7.112)	99
2 (AV)	2048	Temperature heating (1.0.0)	24

NOTE: Maximum amount of Bacnet objects is limited to 150 in homeLYnk.

See chapter 5 for more details.

ARP table

Address Resolution Protocol table is listed in ***Network → ARP table***.

It is used for resolution of network layer addresses into link layer addresses; it converts IP address to a physical address.

Interface	IP address	Mask	MAC address	Flags
eth0	10.154.16.243	*	2c:54:2d:00:da:d0	0x2
eth0	10.154.16.248	*	00:0e:b6:a0:68:a2	0x2
eth0	10.154.20.1	*	00:07:7d:a7:e8:2e	0x2

2015 Schneider Electric. All rights reserved.

KNX connection

KNX specific configuration is located in ***Configurator -> Utilities -> Network→Network → KNX connection*** window.

General

Mode - KNX connection mode. homeLYnk has TP-UART interface by default built-in.

- ***TP-UART*** – Twisted pair connection via black/red plug. Transfer rate 9.6 kB/s.
- ***EIBnet/ IP Tunneling*** – IP connection, minimal 1000x faster than TP-UART. homeLYnk as a server. Unicast, acknowledged data exchange, additional individual address per tunneling connection.
- ***EIBnet/ IP Tunneling (NAT mode) – Network Address Translation mode*** – Allows multiple devices to connect to public network using the same public IPv4 address. It modifies the IP address information in the IPv4 headers while in transit across a traffic routing device.
- ***EIBnet/IP Routing*** – Multicast, unacknowledged data transfer. homeLYnk as a Line or Backbone Coupler.
- ***ACK all group telegrams*** – If homeLYnk communicates directly with another KNX device it must acknowledge received telegrams. Unselect if homeLYnk operates as a sniffer of group addresses only.

KNX address – KNX physical address of the device.

KNX IP features – Use this device with KNX IP features for example, KNXnet/IP network configuration. If not active, then all IP communication from KNX is blocked.

Multicast IP – Multicast IP address.

Multicast TTL – Default value is 1; it allows communication between different sub-networks.

2015 Schneider Electric. All rights reserved.

Maximum telegrams in queue – Count of maximum telegrams in the queue. **NOTE:** If KNX TP is not connected to the device, *Routing* mode should be used in order to ensure the group addresses are updated correctly. Device will reboot after applying the changes.	

2015 Schneider Electric. All rights reserved.

IP >TP filter

Filter accepts or drops received telegrams from the defined KNX devices/physical addresses. All outgoing telegrams are not filtered.

Apply filter to tunneling **–** This filter was created to provide enhanced functionality in comparison to a standard KNX router. Defined filter can be applied even to tunneling mode now, by default it passes all telegrams. This option relates to both directions (IP > TP filter & TP > IP filter)

SRC policy ***[No filter / Accept selected individual addresses / Drop selected individual addresses]*** – Policy to apply to the list of source addresses.

Ind. address list – Lists individual or group addresses. One address per line. Use * (e.g. 1.1.* or 1/1/*) to filter all the addresses in the given line.

DST group policy

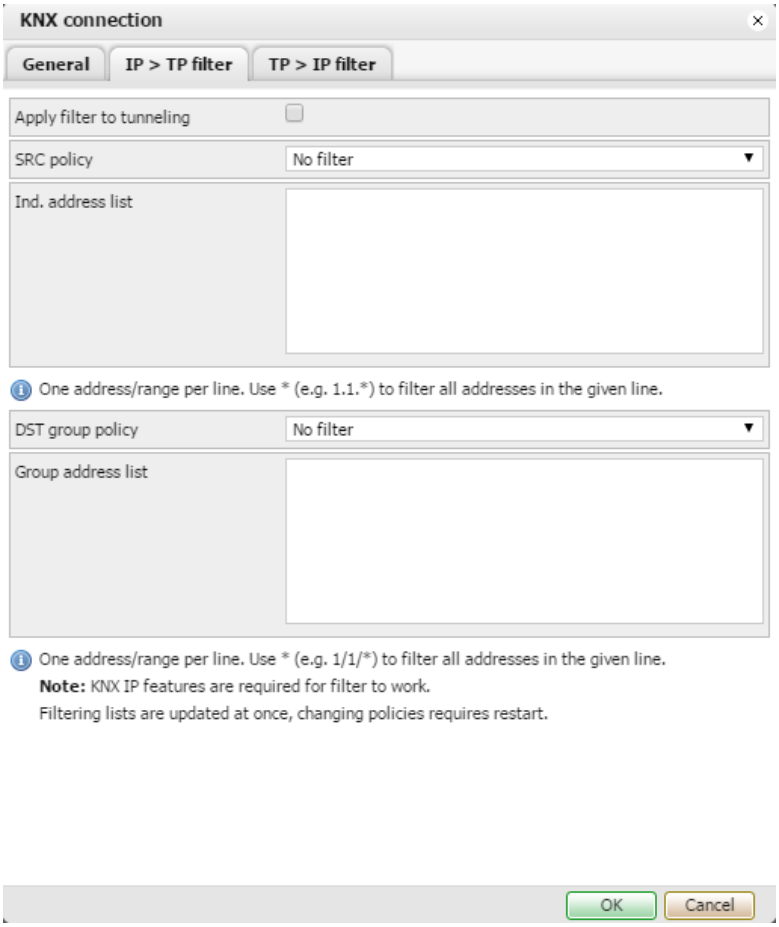
Destination group filter accepts or drops received telegrams belonging to one group as 1/2/3 or subgroup as 1/2/*. All outgoing telegrams are not filtered.

DST group filter ***[No filter / Accept selected individual addresses / Drop selected individual addresses]*** – Policy to apply to the list of destination group addresses.

Group address list – List of group addresses.

One address per line .Use *(e.g. 1/1/*) to filter all the addresses in the given line.

NOTE: KNX IP features should be enabled for filters to work. This applies to incoming telegrams only!


KNX connection
General
IP > TP filter
TP > IP filter
Apply filter to tunneling
SRC policy
No filter
Ind. address list
One address/range per line. Use * (e.g. 1.1.*) to filter all addresses in the given line.
DST group policy
No filter
Group address list
One address/range per line. Use * (e.g. 1/1/*) to filter all addresses in the given line.
Note: KNX IP features are required for filter to work.
Filtering lists are updated at once, changing policies requires restart.
OK
Cancel

2015 Schneider Electric. All rights reserved.

TP >IP filter

Filter accepts or drops received telegrams from the defined KNX devices/physical addresses. All the outgoing telegrams are not filtered.

Apply filter to virtual objects – Virtual object serves for internal data exchange inside homeLYnk (e.g. from Modbus to Visu). If command *gr.update()* is used in Lua, then the group address is not written to TP, but is written to IP only. If this option is ticked, the listed groups are filtered (=not written) from IP and thus virtual.

SRC policy [No filter / Accept selected individual addresses / Drop selected individual addresses] – Policy to apply to the list of source addresses.

Ind. address list – List of individual or group addresses. One address per line. Use * (for example, 1.1.* or 1/1/*) to filter all addresses in the given line.

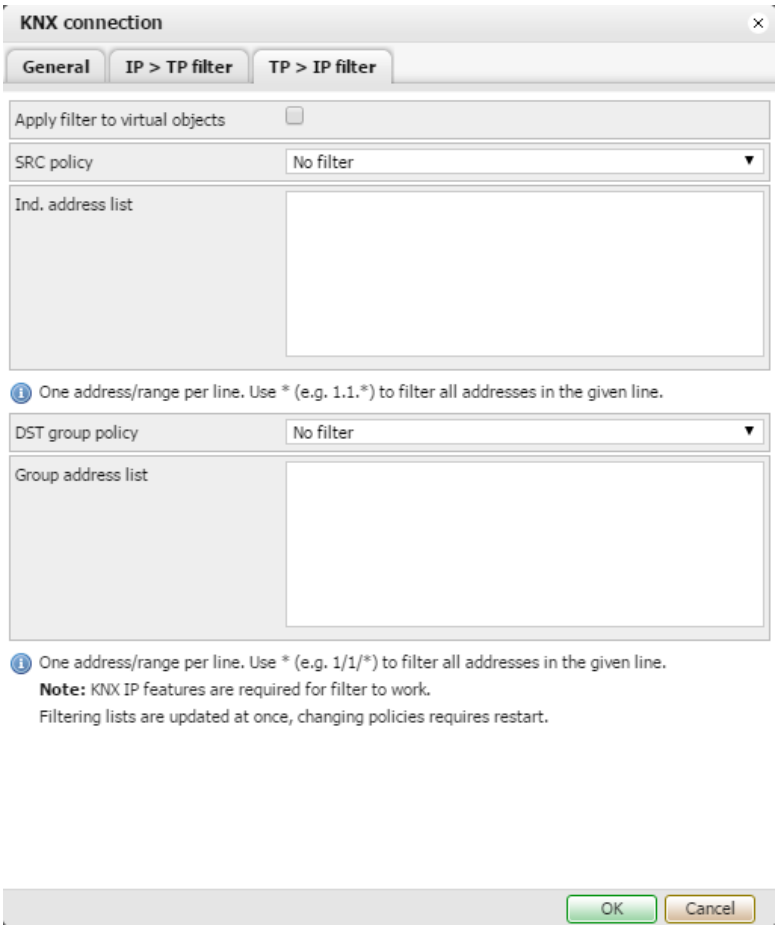
DST group policy – Destination group filter accepts or drops the received telegrams belonging to one group as 1/2/3 or subgroup as 1/2/*. All outgoing telegrams are not filtered.

DST group filter [No filter / Accept selected individual addresses / Drop selected individual addresses] – Policy to apply to the list of the destination group addresses.

Group address list – List of group addresses.

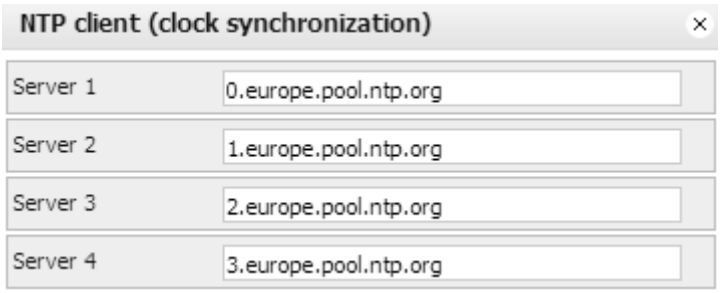
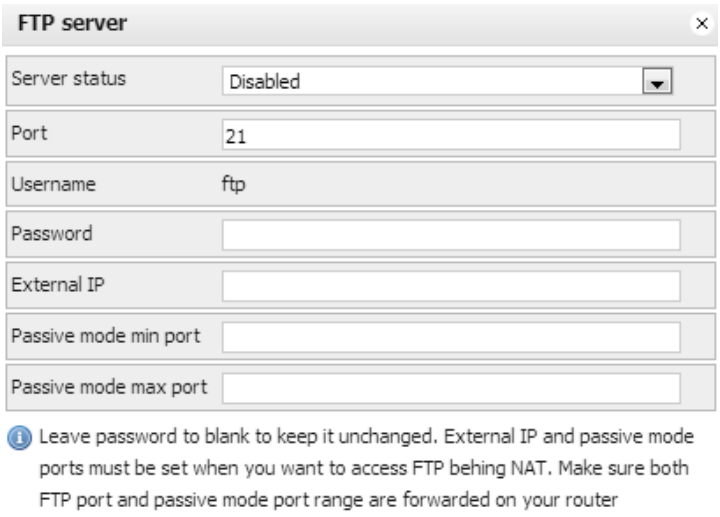
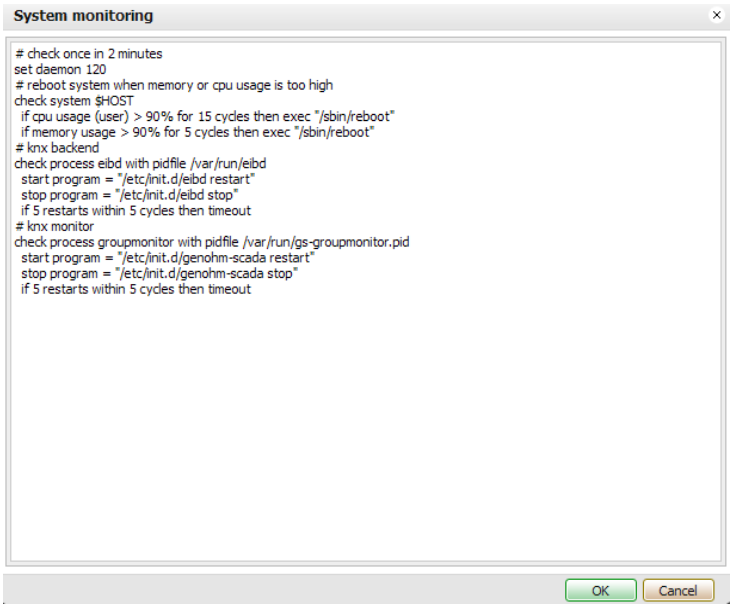
One address per line .Use *(e.g. 1/1/*) to filter all addresses in the given line.

NOTE: KNX IP features should be enabled for filters to work. This applies to the incoming telegrams only!


KNX connection
General
IP > TP filter
TP > IP filter
Apply filter to virtual objects
SRC policy
No filter
Ind. address list
One address/range per line. Use * (e.g. 1.1.*) to filter all addresses in the given line.
DST group policy
No filter
Group address list
One address/range per line. Use * (e.g. 1/1/*) to filter all addresses in the given line.
Note: KNX IP features are required for filter to work.
Filtering lists are updated at once, changing policies requires restart.
OK
Cancel

2015 Schneider Electric. All rights reserved.

3.3.3 System → Services

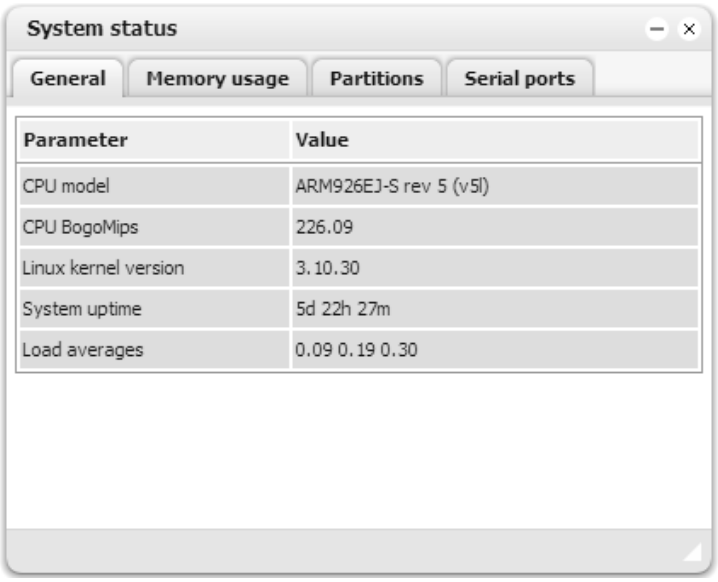
NTP client Network Time Protocol (clock synchronization) Servers 1- 4. Define the server from which date and time is obtained. **NOTE:** Reboot needed.	
FTP server FTP server of homeLYnk can be accessed by enabling ***Service → FTP Server.*** ***Server status*** – Secure tunnel mode. ***Port*** – Port of the service. ***Username*** – Login name, *ftp*. ***Password*** – Password, length 4-20 symbol. Default password is ftp. ***Passive mode min port***– Minimum port for passive mode. ***Passive mode max port*** – Maximum port for passive mode.	
System monitoring Definition of system auto check and auto reboot.	

2015 Schneider Electric. All rights reserved.

3.3.4 System → Status

System status

System information is shown is the following tabs:



General

Information about hardware provided by kernel.

Memory usage

Current memory used by the system.

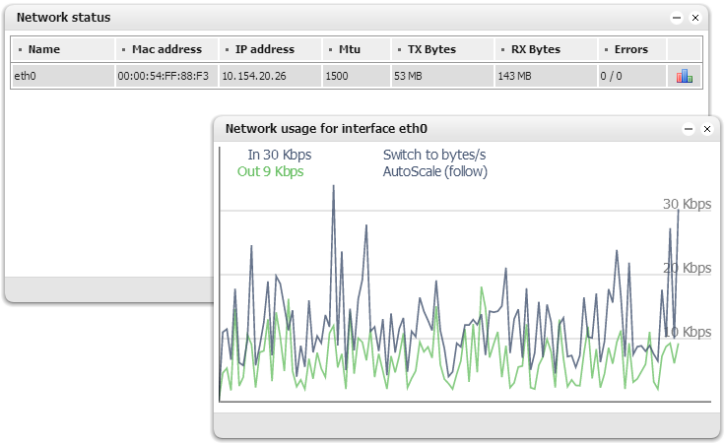
Partitions

List of partitions available in the system.

Serial ports

List of serial ports available in the system.

Network status



2015 Schneider Electric. All rights reserved.

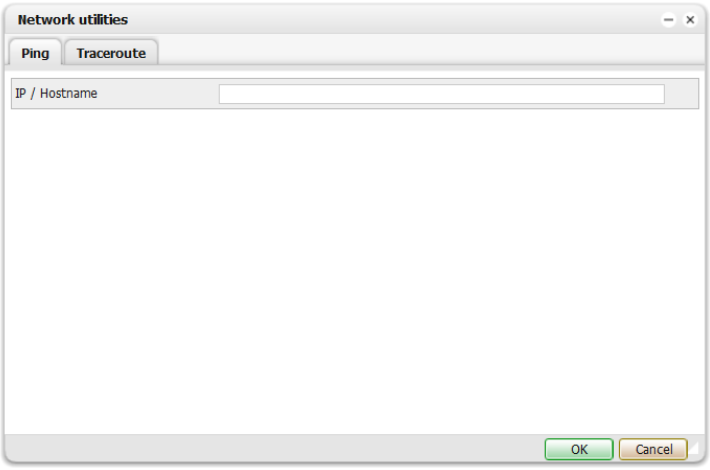
Network utilities

Ping

The Computer network tool id used to test whether a particular host is reachable across an IP network.

Trace route

The computer network diagnostic tool is used for displaying the route (path) and measuring transit delays of packets across an Internet Protocol (IP) network.



System log

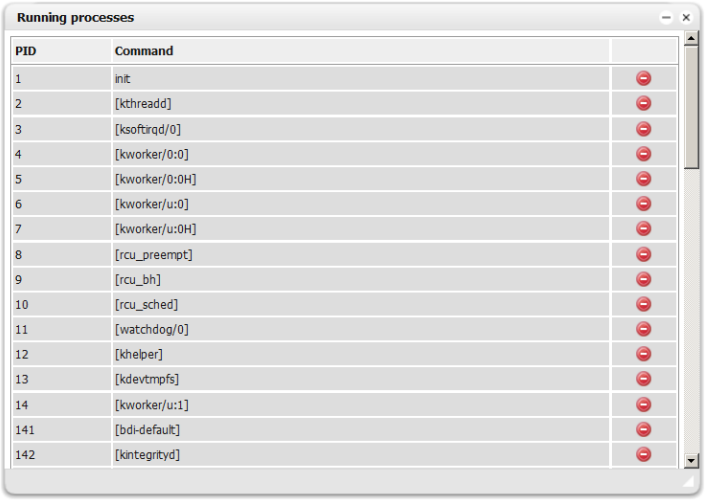
Log entries

Log files are automatically created and maintained by homeLYnk of all the activities performed.

System log
Log entries
May 29 13:02:48 spaceLYnk daemon.info avahi-daemon[818]: Service "Web Server on spaceLYnk" (/etc/avahi/services/http.service)
May 29 13:02:47 spaceLYnk daemon.info avahi-daemon[818]: Server startup complete. Host name is spaceLYnk.local. Local service
May 29 13:02:46 spaceLYnk daemon.info avahi-daemon[818]: Registering HINFO record with values 'ARMV5TEJL'/'LINUX'.
May 29 13:02:46 spaceLYnk daemon.info avahi-daemon[818]: Registering new address record for 192.168.0.10 on eth0.IPv4.
May 29 13:02:46 spaceLYnk daemon.info avahi-daemon[818]: Network interface enumeration completed.
May 29 13:02:46 spaceLYnk daemon.info avahi-daemon[818]: New relevant interface eth0.IPv4 for mDNS.
May 29 13:02:46 spaceLYnk daemon.info avahi-daemon[818]: Joining mDNS multicast group on interface eth0.IPv4 with address
May 29 13:02:46 spaceLYnk daemon.info avahi-daemon[818]: Loading service file /etc/avahi/services/http.service.
May 29 13:02:46 spaceLYnk daemon.warn avahi-daemon[818]: WARNING: No NSS support for mDNS detected, consider installing nss-

2015 Schneider Electric. All rights reserved.

Running processes



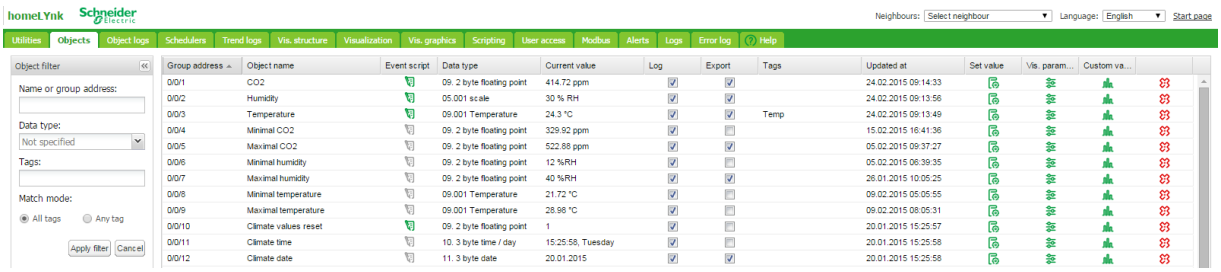
2015 Schneider Electric. All rights reserved.

3.4 Objects

List of KNX network objects appear in the ***Objects*** menu. The object is listed accordingly:

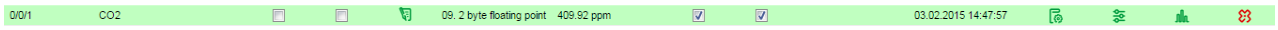
1. Captured by sniffing the bus for telegrams from unknown group addresses (if enabled in ***Utilities***).
2. Added manually.
3. Importing ESF file (in ***Utilities***).

0bjects are sorted with the following parameters– ***Group address, Object name, IP>TP filter, TP>IP filter, Event script, Data type, Current value, Log, Export, Tags, Updated at, Set value, Vis.parameters*** and ***Custom values***.

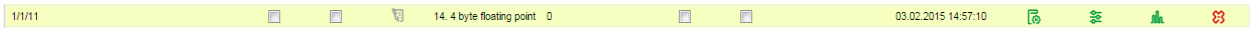


Objects are further distinguished by color of their background for quick overview:

- Green – Object value actually updated.



- Yellow – Object actually discovered by a bus sniffer.



2015 Schneider Electric. All rights reserved.

3.4.1 Object parameters

Object name – Name for the object.

Group address – Group address of this object.

Data type – KNX data type for the object. This has to be set once the homeLYnk sniffs the new object for actual work.

Current value– Actual value of the object.

Tags – Assigns this object to some tag which can be later used in the writing scripts, for example, *All_lights_first_floor*. (Please refer to the Script library for use cases).

Unit/suffix – Add unit/suffix to value of object. Units which cannot be created from keyboard can be created in external editor and pasted into the browser.

Log – Enable logging for this object. Logs appear in the ***Object logs*** tab.

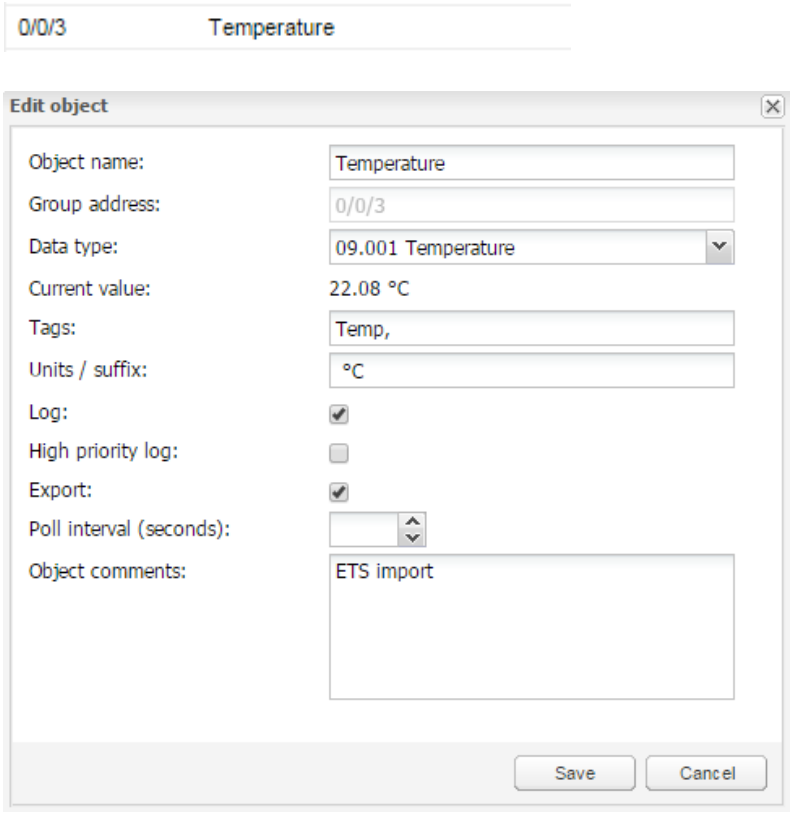
High priority log – This option shifts high priority logs up on the screen listing (tab *Logs*). If defined limit of logs is exceeded, low priority logs at the end of listing are deleted first. This function secures that high importance logs stay visible for a longer period of time.

Export – Makes object visible by remote XML requests.

Poll interval (seconds) – Performs automatic object read after the selected time interval.

Object comments –Object's further description. Can be also used for filtering.

To change the settings for existing or new objects, click on the specific list entry address or name.



3.4.2 Event Script

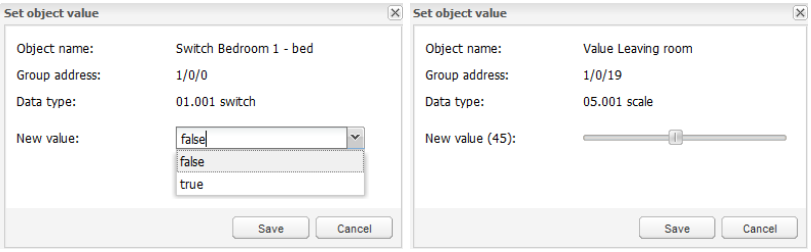
By pressing button in the Object list, the Script editor will open and the event based script can be created. Script will run each time the telegram is sent to the selected group. If the script is attached to a group, icon changes to green .

2015 Schneider Electric. All rights reserved.

3.4.3 Set object value

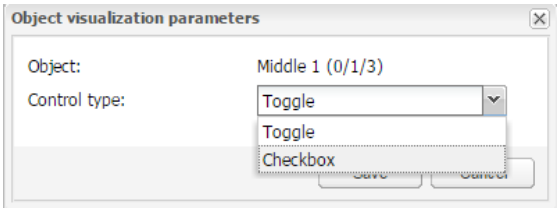
In the object list, by pressing on the button, the state of the object can be changed.

The appearance of the ***New value window*** depends on what the visualization parameters are set for specific objects.



3.4.4 Object Visualization Parameters

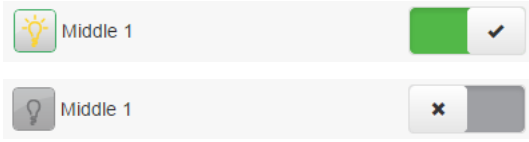
By pressing on the button , the corresponding object specific visualization parameters for this type can be set.



<u>1-bit</u>

Control Type – Types of the visual control element:

- ***Toggle***
- ***Checkbox***



<u>4-bit (3-bit controlled)</u>

Step size – Step size example for blinds control:

2-bit (1-bit controlled), 1-byte unsigned integer (scale), 1-byte signed integer, 2-byte unsigned integer, 2-byte signed integer, 2-byte floating point (temperature), 4-byte unsigned integer, 4-byte signed integer, 4-byte floating point.

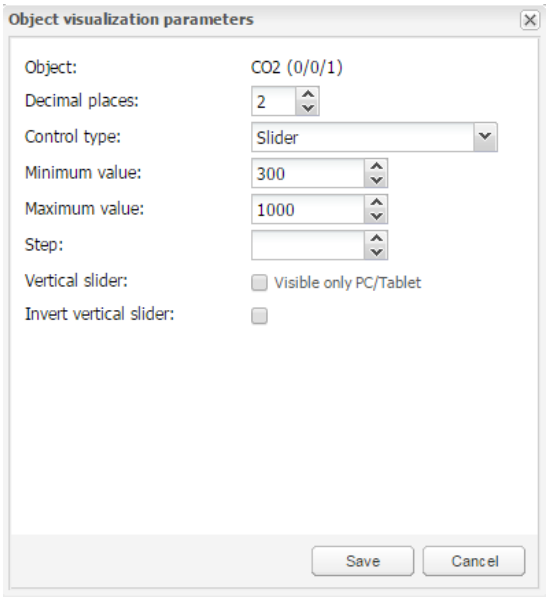
Control type – Types of the visual elements:

Direct +/-

Slider

Circular slider

Custom value select



Minimum value – Define minimum value for visualization only.

Maximum value – Define maximum value for visualization only.

Step – If defined, value changes depending on the defined step.

2015 Schneider Electric. All rights reserved.

Vertical slider – Vertical position of slider e.g. for Blinds control.

Invert vertical slider – Inverts direction of vertical slider.

Slider color – Defines slider filling color.

Background color – Defines background of circular slider.

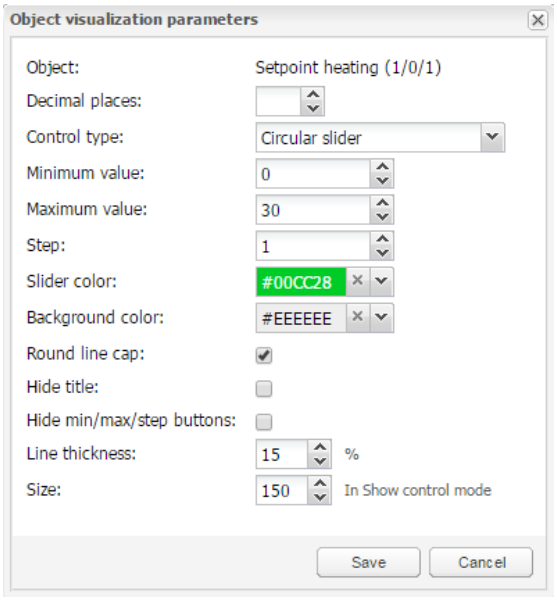
Round line cap – Rounding edges of circular slider.

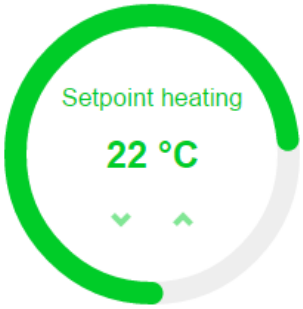
Hide title – Hide Object/custom name in visualization.

Hide min/max/step buttons – Hide buttons in visualization for touch screen enabled devices.

Line thickness – Thickness of circular slider control line (1-50%).

Size – Size of circular slider (150-500 pixels).





2015 Schneider Electric. All rights reserved.

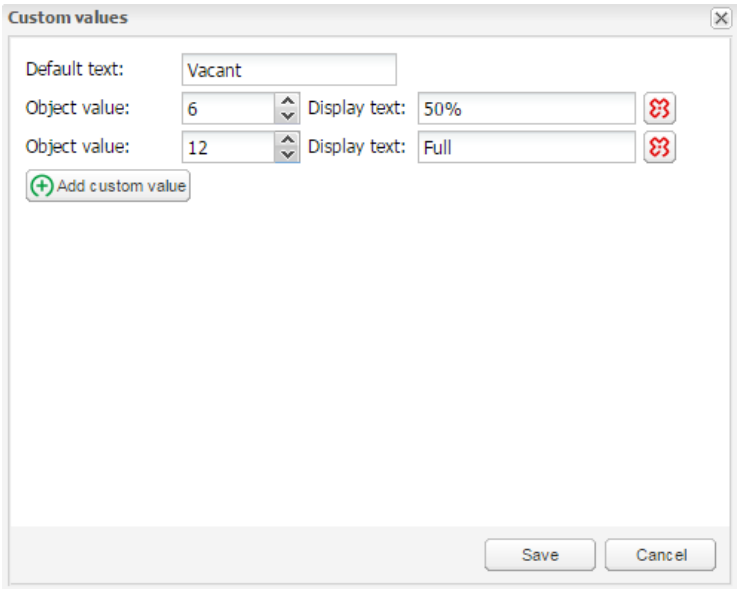
3.4.5 Custom Text Value

In the object list, by pressing the button, custom text can be added to the object values.

Custom text values can be set only to Boolean or integer values.

Default text – Text displayed if value is not defined.

Object value – ***Add custom value***, select ***Object value*** and define ***Display*** *text*.



Sample of displayed value from above settings in visualization.

3.4.6 Object Control Bar

Add new object – Manually add new objects to the list.

Auto update enabled – Specifies either the object list is updated automatically or not.

Clear – Clear the list of filtered group addresses.

Mass edit – For mass edit of filter selected objects.

Next/Previous page – Move to the next or previous page.

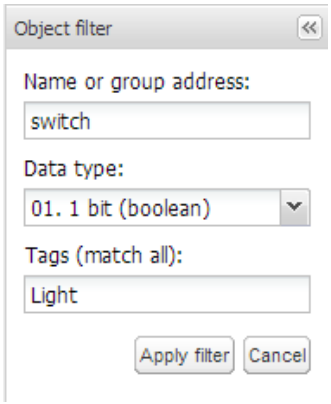
Refresh – Refresh the object list.

TP>IP policy – Selected filtering policy.

2015 Schneider Electric. All rights reserved.

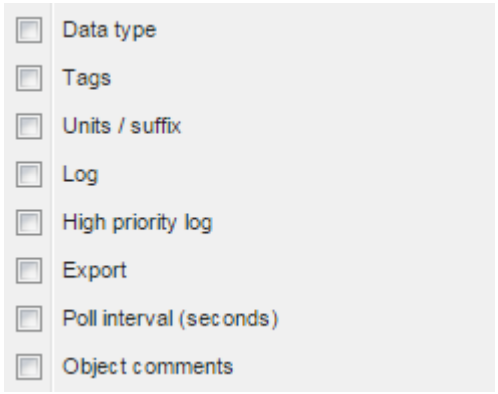
3.4.7 Object filter

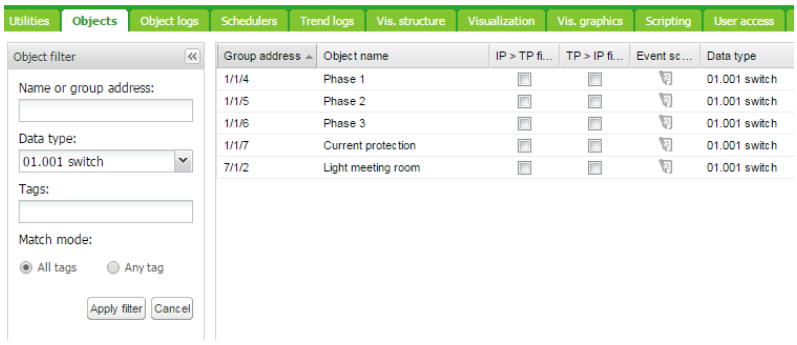
On the left side of the object list, you can filter. To perform the Object filter, enter the Name or group address, tag or specify the Data type of the object and press ***Apply filter*** button.



3.4.8 Mass edit

Objects filtered in ***Object filter*** can be mass edited by

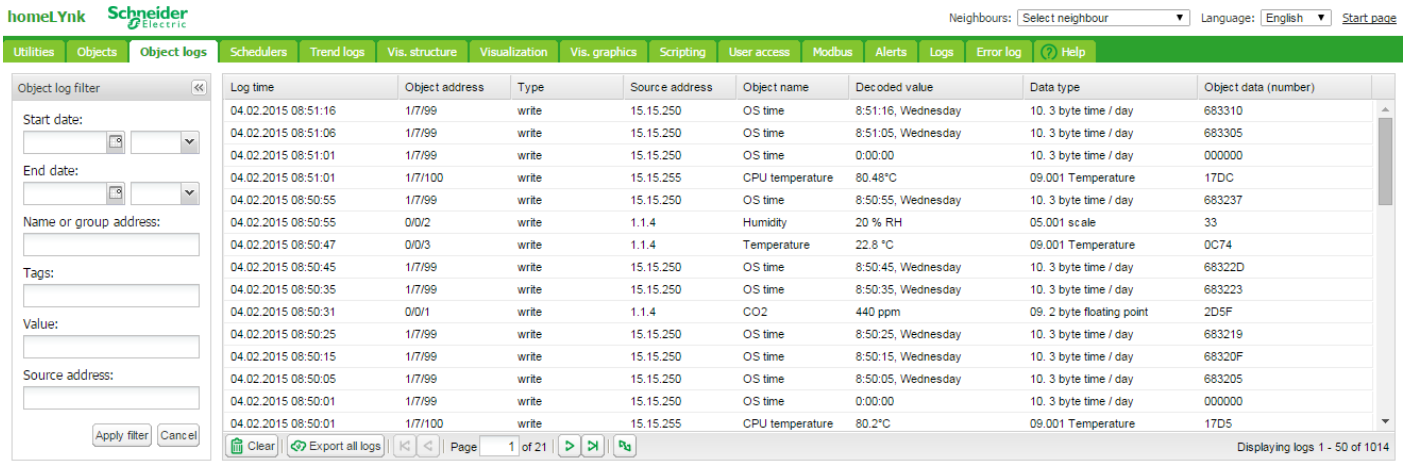




2015 Schneider Electric. All rights reserved.

3.5 Object Logs

Object's historical telegrams are available in ***Object logs*** tab. After logging is enabled for object, all the historical and future data will be logged in.


homeLYnk
Schneider Electric
Neighbours: Select neighbour
Language: English
Start page
Utilities
Objects
Object logs
Schedulers
Trend logs
Vis. structure
Visualization
Vis. graphics
Scripting
User access
Modbus
Alerts
Logs
Error log
Help
Object log filter
Start date:
End date:
Name or group address:
Tags:
Value:
Source address:
Apply filter
Cancel
Log time | Object address | Type | Source address | Object name | Decoded value | Data type | Object data (number)
04.02.2015 08:51:16 | 1/7/99 | write | 15.15.250 | OS time | 8:51:16, Wednesday | 10. 3 byte time / day | 683310
04.02.2015 08:51:06 | 1/7/99 | write | 15.15.250 | OS time | 8:51:05, Wednesday | 10. 3 byte time / day | 683305
04.02.2015 08:51:01 | 1/7/99 | write | 15.15.250 | OS time | 0:00:00 | 10. 3 byte time / day | 000000
04.02.2015 08:51:01 | 1/7/100 | write | 15.15.255 | CPU temperature | 80.48°C | 09.001 Temperature | 17DC
04.02.2015 08:50:55 | 1/7/99 | write | 15.15.250 | OS time | 8:50:55, Wednesday | 10. 3 byte time / day | 683237
04.02.2015 08:50:55 | 0/0/2 | write | 1.1.4 | Humidity | 20 % RH | 05.001 scale | 33
04.02.2015 08:50:47 | 0/0/3 | write | 1.1.4 | Temperature | 22.8 °C | 09.001 Temperature | 0C74
04.02.2015 08:50:45 | 1/7/99 | write | 15.15.250 | OS time | 8:50:45, Wednesday | 10. 3 byte time / day | 68322D
04.02.2015 08:50:35 | 1/7/99 | write | 15.15.250 | OS time | 8:50:35, Wednesday | 10. 3 byte time / day | 683223
04.02.2015 08:50:31 | 0/0/1 | write | 1.1.4 | CO2 | 440 ppm | 09. 2 byte floating point | 2D5F
04.02.2015 08:50:25 | 1/7/99 | write | 15.15.250 | OS time | 8:50:25, Wednesday | 10. 3 byte time / day | 683219
04.02.2015 08:50:15 | 1/7/99 | write | 15.15.250 | OS time | 8:50:15, Wednesday | 10. 3 byte time / day | 68320F
04.02.2015 08:50:05 | 1/7/99 | write | 15.15.250 | OS time | 8:50:05, Wednesday | 10. 3 byte time / day | 683205
04.02.2015 08:50:01 | 1/7/99 | write | 15.15.250 | OS time | 0:00:00 | 10. 3 byte time / day | 000000
04.02.2015 08:50:01 | 1/7/100 | write | 15.15.255 | CPU temperature | 80.2°C | 09.001 Temperature | 17D5
Clear
Export all logs
Page 1 of 21
Displaying logs 1 - 50 of 1014

Filtering is available when there is a need to find specific period information:

- ***Start date*** – Start date and time for log filtering
- ***End date –*** Start date and time for log filtering
- ***Name or group address*** – Specific name or group address of the object
- ***Tags*** – Group objects with the same tags filtered
- ***Value*** – Specific object value
- ***Source address*** – Specific source address

All logs can be cleared by pressing the ***Clear*** button.

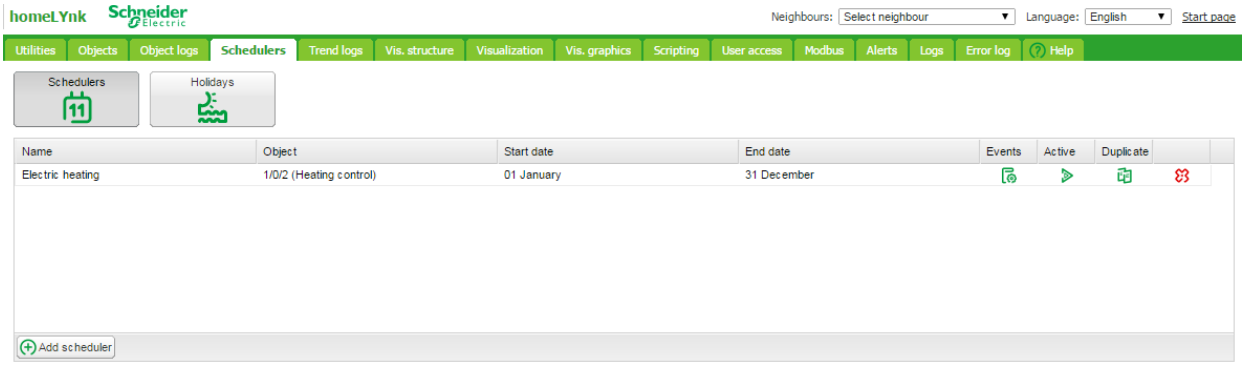
For important objects, activate the parameter ***High Priority log*** together with ***Log*** parameter. This function will list the selected objects on the top of the ***Object logs*** list.

NOTE: Logging memory is set up in the ***Utilities → Configurations.***

2015 Schneider Electric. All rights reserved.

3.6 Schedulers

Schedulers allow the end user to control KNX group address values based on the date or day of the week.



3.6.1 Add new scheduler

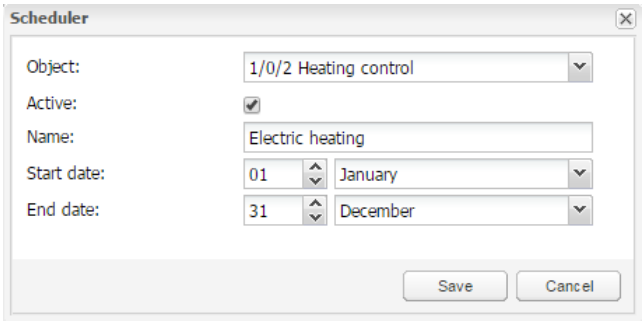
Object – The object group address which will be controlled by the scheduler.

Active – Defines whether a scheduler is active or not.

Name – Name of the scheduler.

Starts date – Start date of the scheduler.

End date – End date of the scheduler.



2015 Schneider Electric. All rights reserved.

3.6.2 Scheduler Events

Event can be added both in the administrator interface and by the end user in the ***special User mode schedulers*** interface.

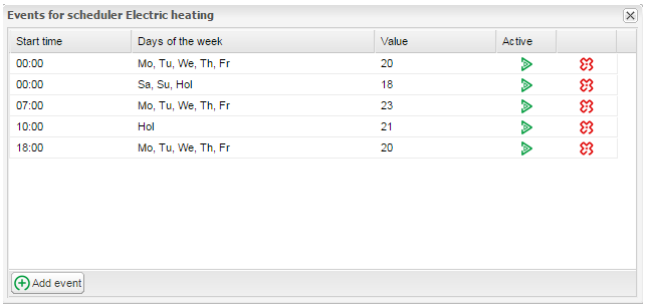
Active – Defines the event to be active or not.

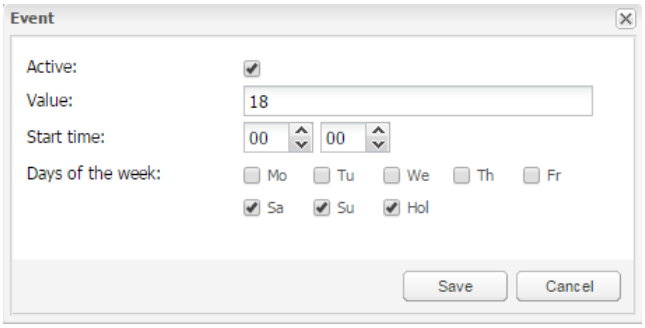
Value – Value to send to the group address when the event will be triggered.

Start time – Start time for the event.

Days of the week – Days of the week when the event will be triggered.

Hol – Holidays which are defined in the **Holidays** tab.

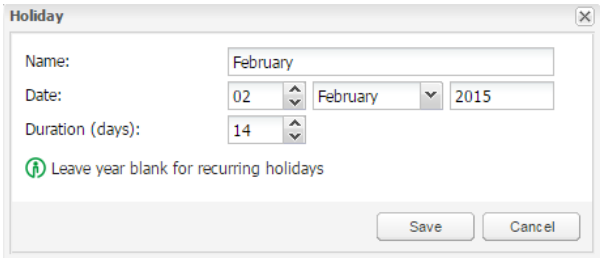




3.6.3 Scheduler Holidays

Once the event will be marked to run on ***Hol***, Holiday entries will be activated.

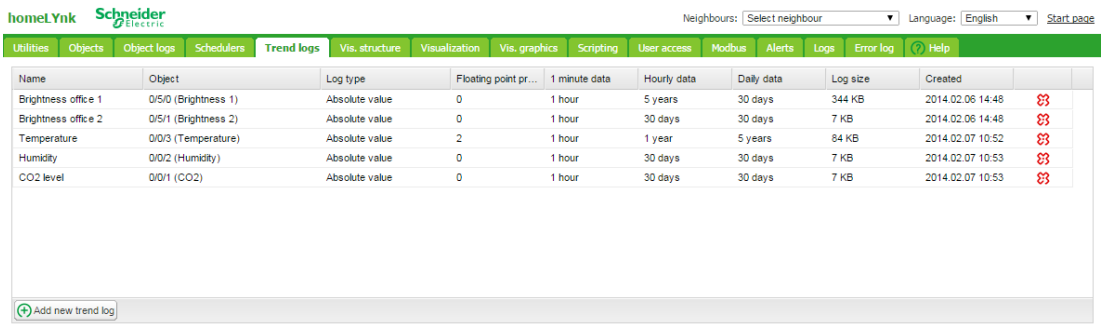
NOTE: ***Scheduler*** visualization is not supported in the ***Smartphone visualization***.

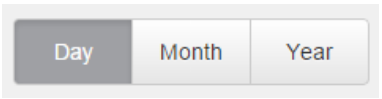


2015 Schneider Electric. All rights reserved.

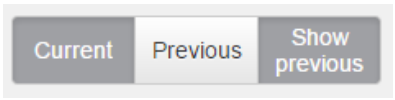
3.7 Trend logs

Trend logs or so called data logging allows the end user to store the selected data and compare the different time periods from the past.





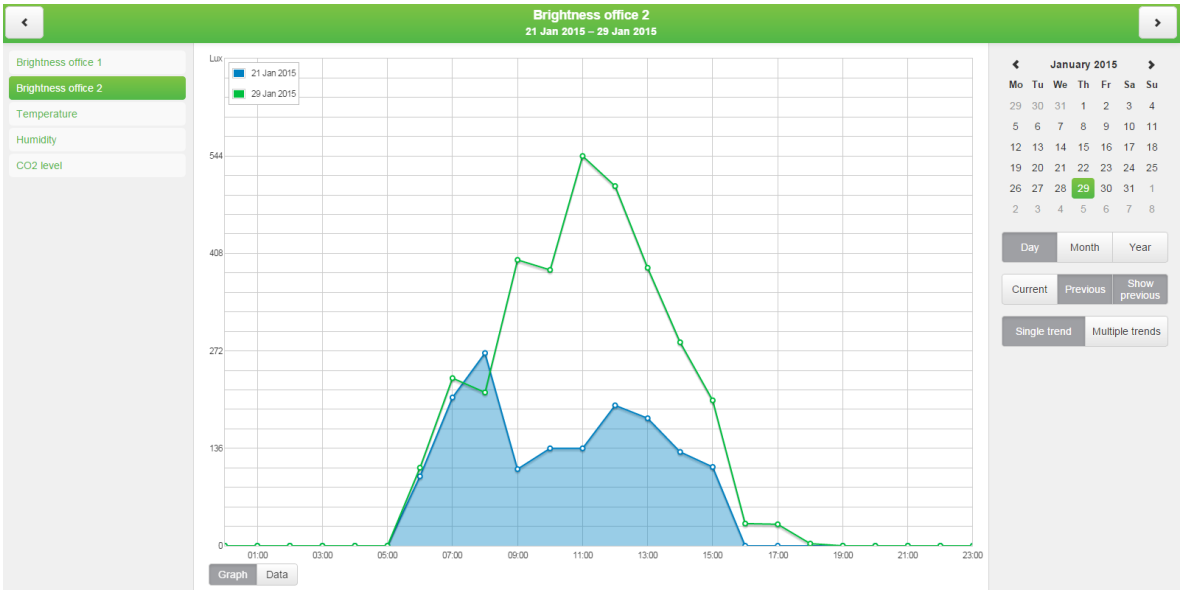
Selecting period of trend(s).



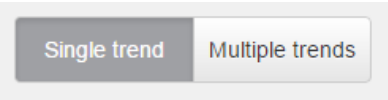
Show previous – enable/disable function of previous values for selected time period (Day/Month/Year) for data comparation.

Current – for selection of current date.

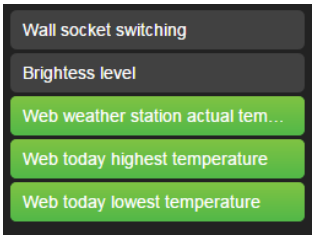
Previous – for selection of previous date.



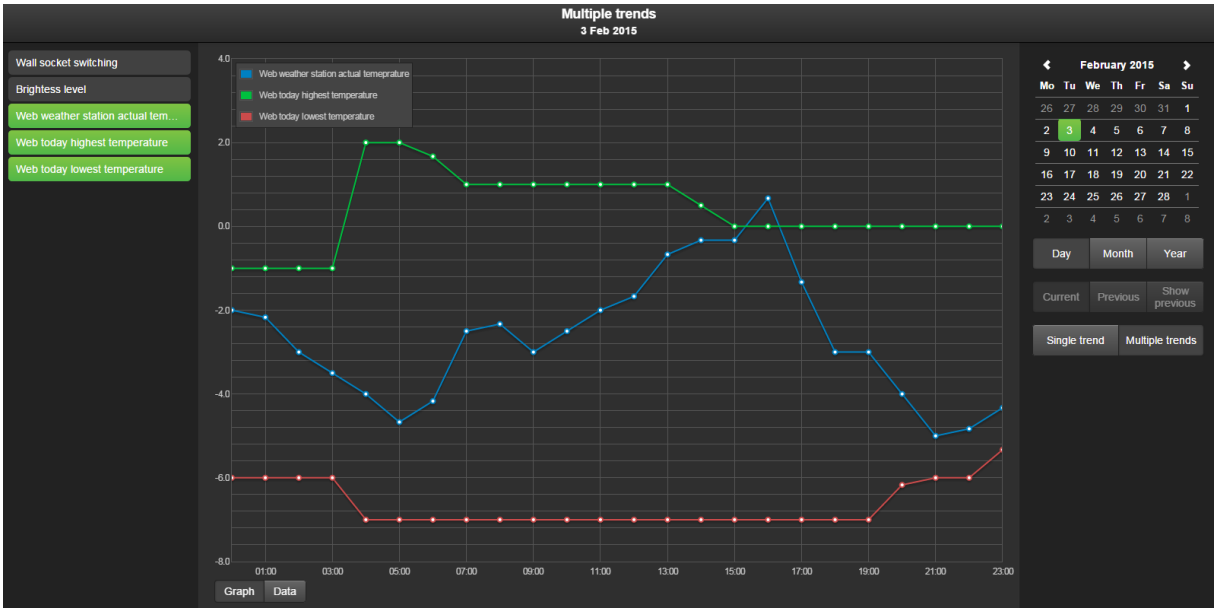
2015 Schneider Electric. All rights reserved.



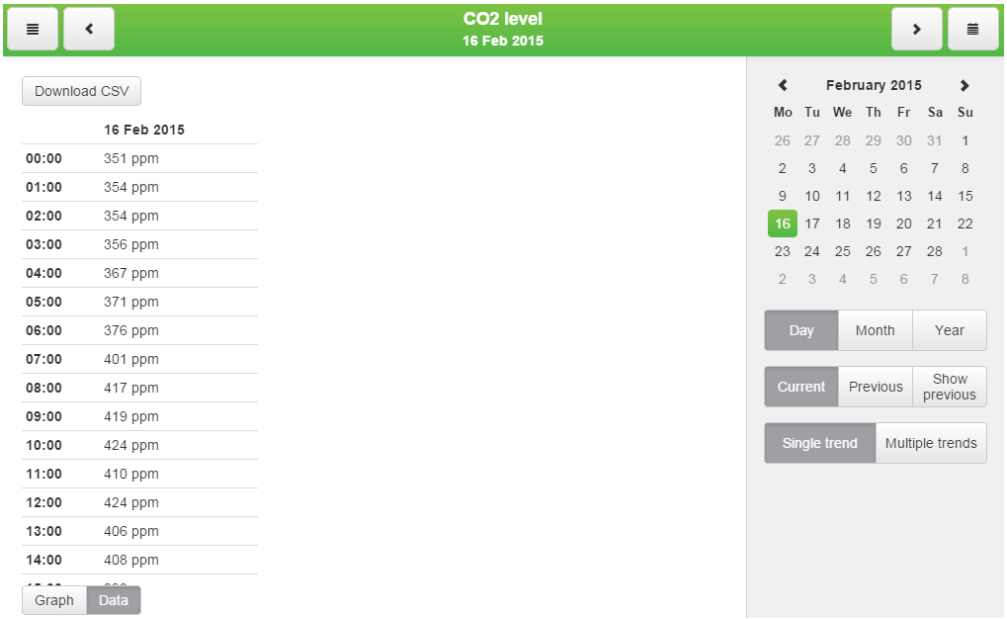
Selection between displaying single and multiple trends (in dark theme).



Selection of visible trends is done in Trends list.



Data can be also displayed in numeric format and exported in CSV format for further use.



2015 Schneider Electric. All rights reserved.

3.7.1 Add New Trend Log

Object – Choose from the list of objects the one to make the trends for.

Name – Name of the trend.

Log type – Type of the log.

Counter - Used to count the data.

Absolute value – Saves the actual readings.

Floating point precision – If the object is floating type, then the precision needs to be selected. Example, 1.1111 = precision is 4.

1 minute data – Average value of 1 minute for specific time interval data will be shown on the trend. Example, if 1 hour – trend step will be 1 hour with average 60 readings data.

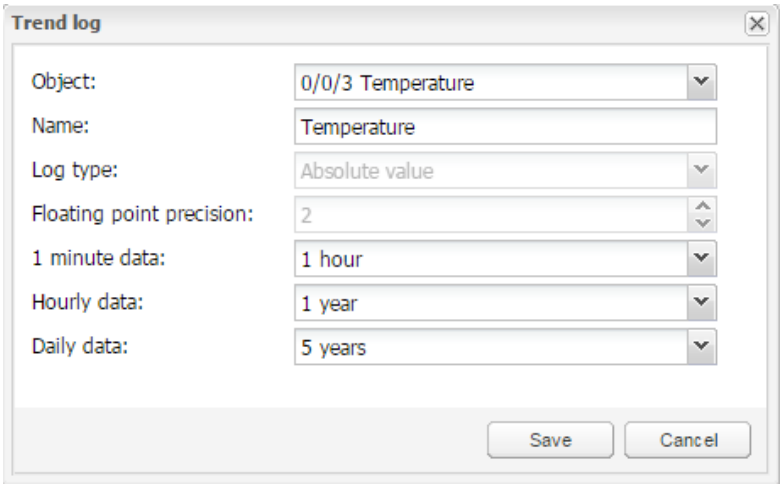
Hourly data – Time of storage of hourly data. (Max.5 years.)

Daily data – Time of storage of daily data. (Max. 10 years).

NOTE: Trend logs are stored in internal SD card memory.

NOTE: Please note that if the log type is set to ***Counter*** it cannot have permanently decreasing tendency. For this option use ***Counter with negative delta.***

Trends visualization is not supported in the ***Smartphone visualization***.



2015 Schneider Electric. All rights reserved.

3.8 Vis. Structure

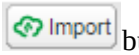
Vis. Structure is used for creating all building levels and visualizations plans. Additionally, it can create ***Layouts*** and ***Widgets*** for the plans visualization.

Starting new project, only ***Layout*** and ***Widget*** folders are visible. Adding new level, allows the end user to define specific ***Plan*** of the flat. ***Layouts*** and ***Widgets*** are additional tools which are not mandatory for basic visualizations; they can be defined and implemented in other ***Plans***.

3.8.1 Levels

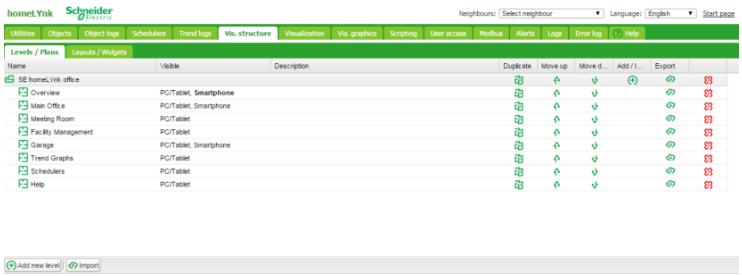
To add new Level, press

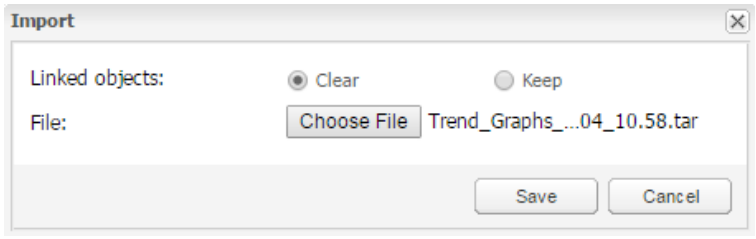
button.
Main level usually is the project name.
Additional levels can be added later.

To import Level press 

button.

Imported level can be added or replace the original plan.

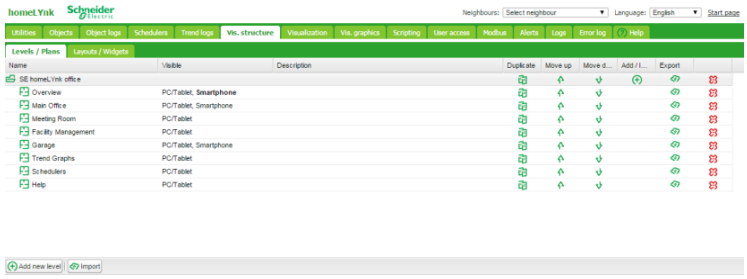




3.8.2 Second level

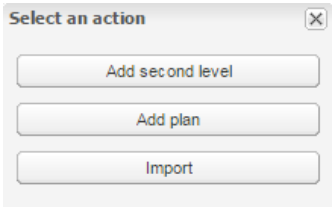
Second level is used in buildings with multiple floors.

If you need additional levels press button next to the main level.



Select ***Add second level*** and give it a name and sort order.

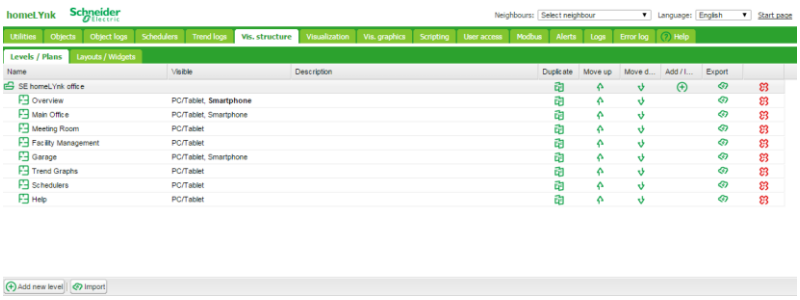
Each level can be duplicated or imported together with sublevels and plans by pressing the duplicate icon next to the level.



2015 Schneider Electric. All rights reserved.

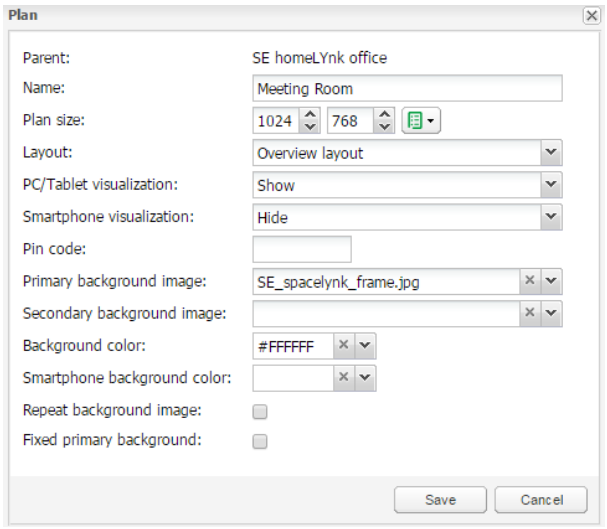
3.8.3 Plan

Plan can show either one room in a flat with cumulated functions or one function (as lighting or heating) of the whole flat.o add Plans press button next to a level under which the plan is to be added and select ***Add plan***.



Name – Name for the plan.

Plan size – Size of the plan. For pre-defined sizes pop-up menu .



Layout – Layout for this specific plan. All objects from Layout will be duplicated on this particular plan, including the background color and the plan image. If they are not defined separately for this specific plan. Layout should be created before adding the Plan.

PC/Tablet visualization

[Show, Show and make default, Hide] – Visibility for this particular plan in the PC/Tablet visualization.

Smartphone visualization

[Show, Show and make default, Hide] – Visibility for this particular plan in the Smartphone visualization.

Pin code – Possibility to protect each plan with Pin code.

Primary background image – Choose the primary background of the plan.

Secondary background image – Choose the secondary background of the plan for parallax look of the visualization. Select background previously added to ***Vis. graphics -> Images/Backgrounds*****.**

Background color – Choose the background color of the plan.

Smartphone background color – Choose the background color of the plan for Smartphone

2015 Schneider Electric. All rights reserved.

visualization.

Repeat background image – Either to show the image once, or repeat it and fill the whole plan.

Fixed primary background – Static primary picture in Parallax projection.

Each Plan can be duplicated together with all the components on a plan by pressing the duplicate icon next to the plan .

NOTE: Content of the created Plan should be defined under the Visualization tab.

Empty plan (no objects) will be not visible in visualization.

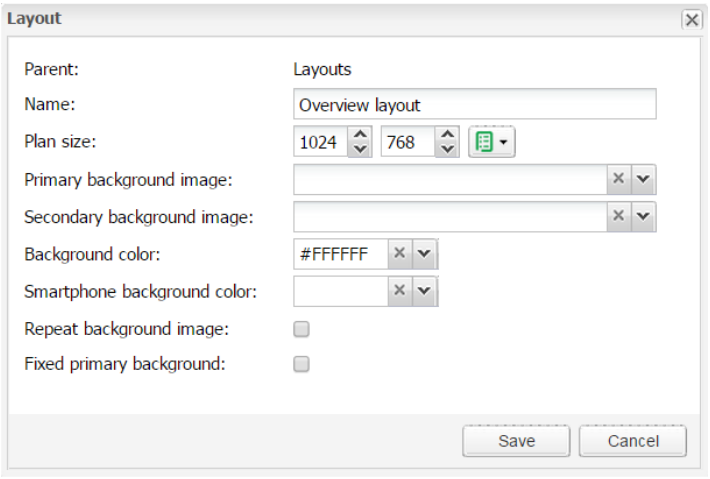
3.8.4 Layout

Layout is advanced background for plans. Any object from the editor can be placed on the layout which later can be attached to one or many plans. All objects from the layout will be visible on the plan, but all the objects on the plan will be above the objects from the layout.

To add Layout press button next to a Layout folder or Add new layout button.

Each Layout can be duplicated together with all the components by pressing the duplicate icon next to the Layout .

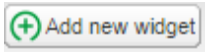
NOTE: Content of this layout should be defined under the Visualization tab.

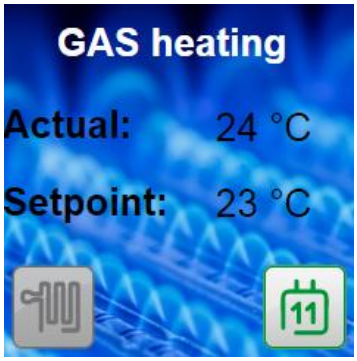


2015 Schneider Electric. All rights reserved.

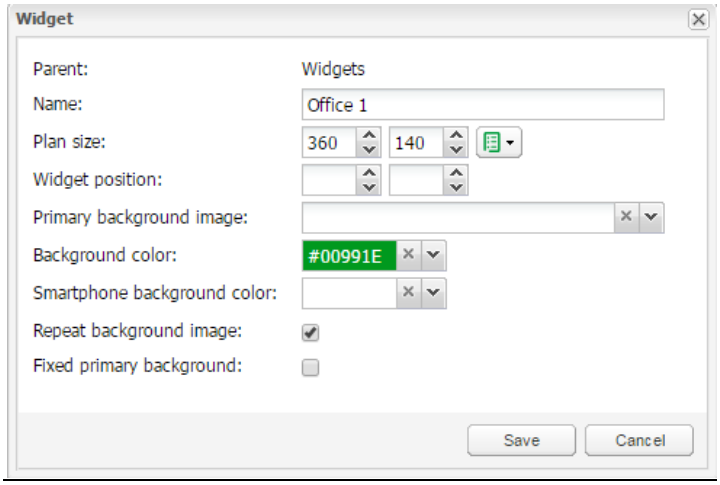
3.8.5 Widget

Widget is a small web page which can be attached to a button and pop-up when activated.

To add the widgets press button next to the widgets folder or  button. Each widget can be duplicated together with all the components by pressing the duplicate icon next to the widget .



NOTE: Content of this widget should be defined under the Visualization tab.



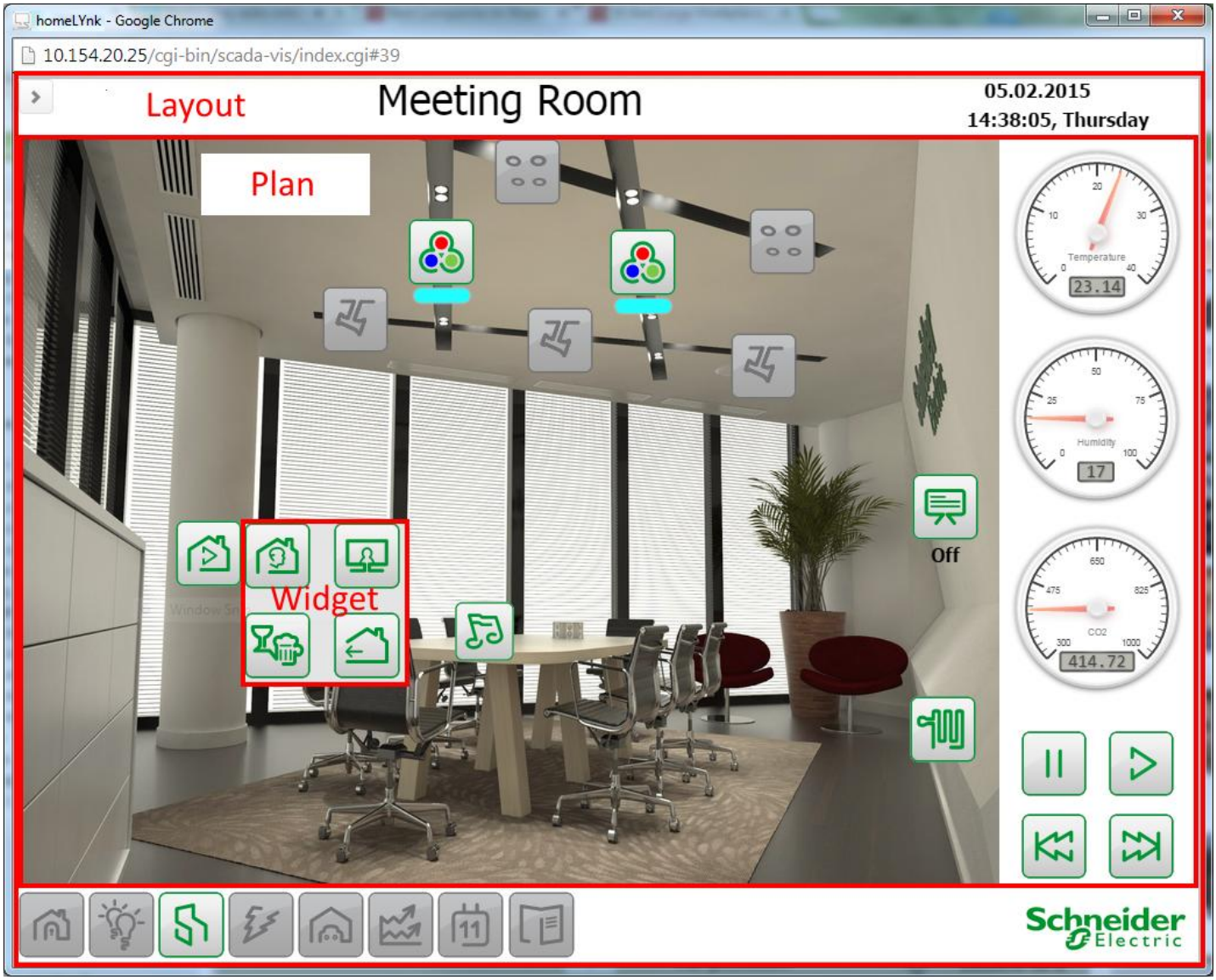
Widget size always has to be smaller than the plan on which it is placed on.

Empty widget (no objects) will be not visible in visualization.

2015 Schneider Electric. All rights reserved.

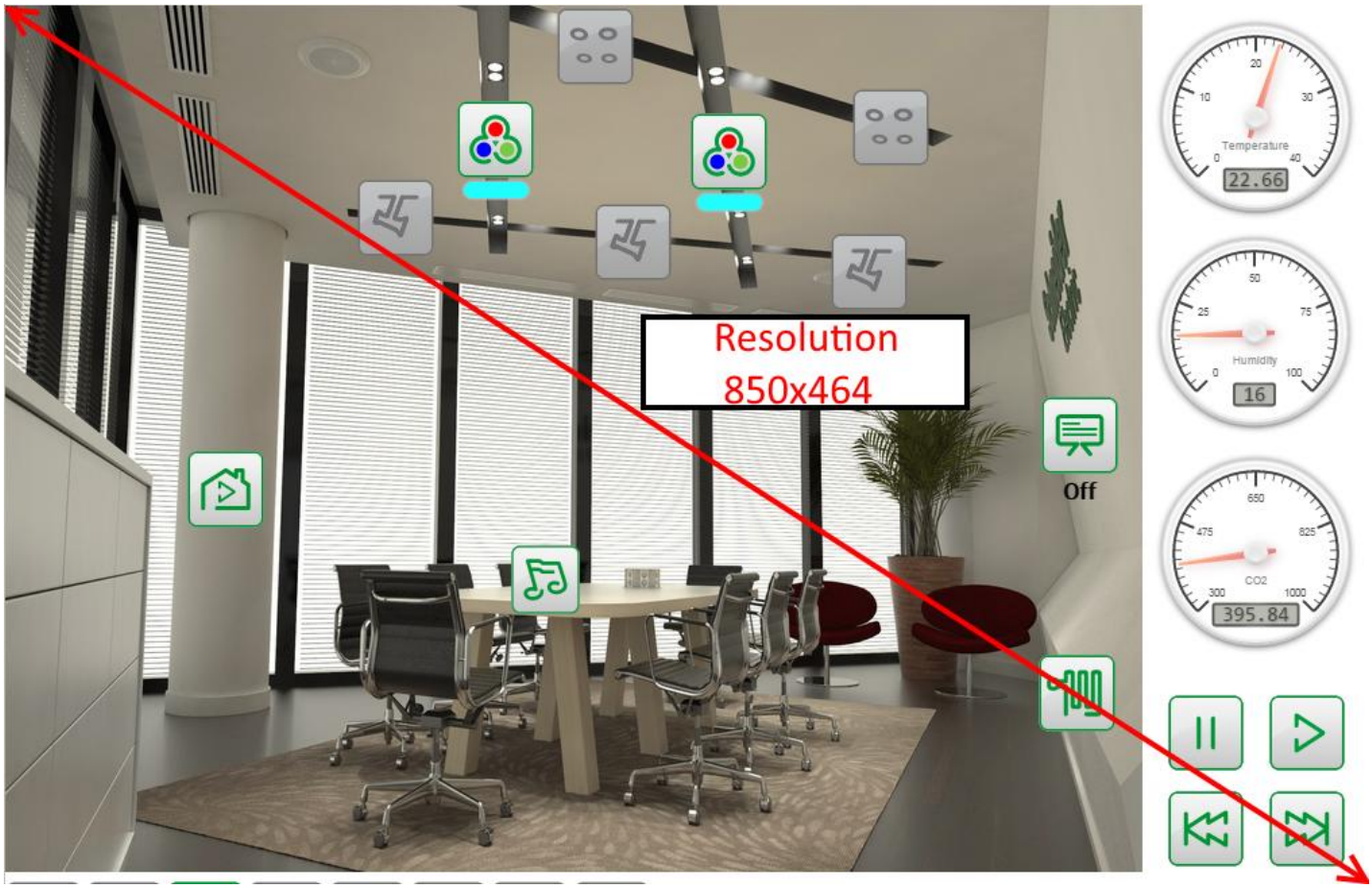
3.8.6 Visualization Structure Example

3.8.7 Plan

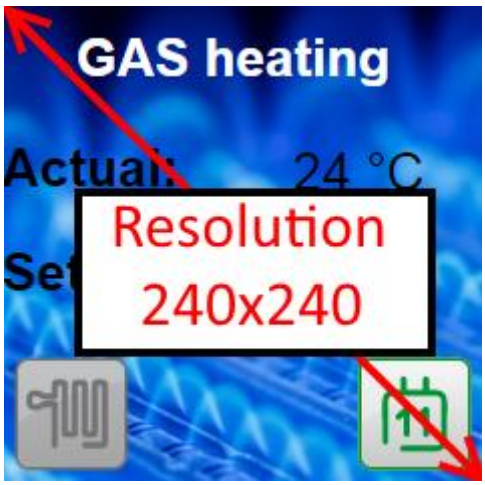


2015 Schneider Electric. All rights reserved.

3.8.8 Layout



3.8.9 Widget



2015 Schneider Electric. All rights reserved.

3.8.10 Visualization Object Order

Each object on visualization, has its priority which is described from the highest to the lowest order:

1. Object on plan
2. Plan link as text on plan
3. Plan link as icon on plan
4. Camera on plan
5. Graph on plan
6. Gauge on plan
7. Text label on plan
8. Image on plan
9. Frame on plan
10. Object on layout
11. Plan link as text on layout
12. Plan link as icon on layout
13. Camera on layout
14. Graph on layout
15. Gauge on layout
16. Text label on layout
17. Image on layout
18. Frame on layout
19. Background of plan
20. Background of layout

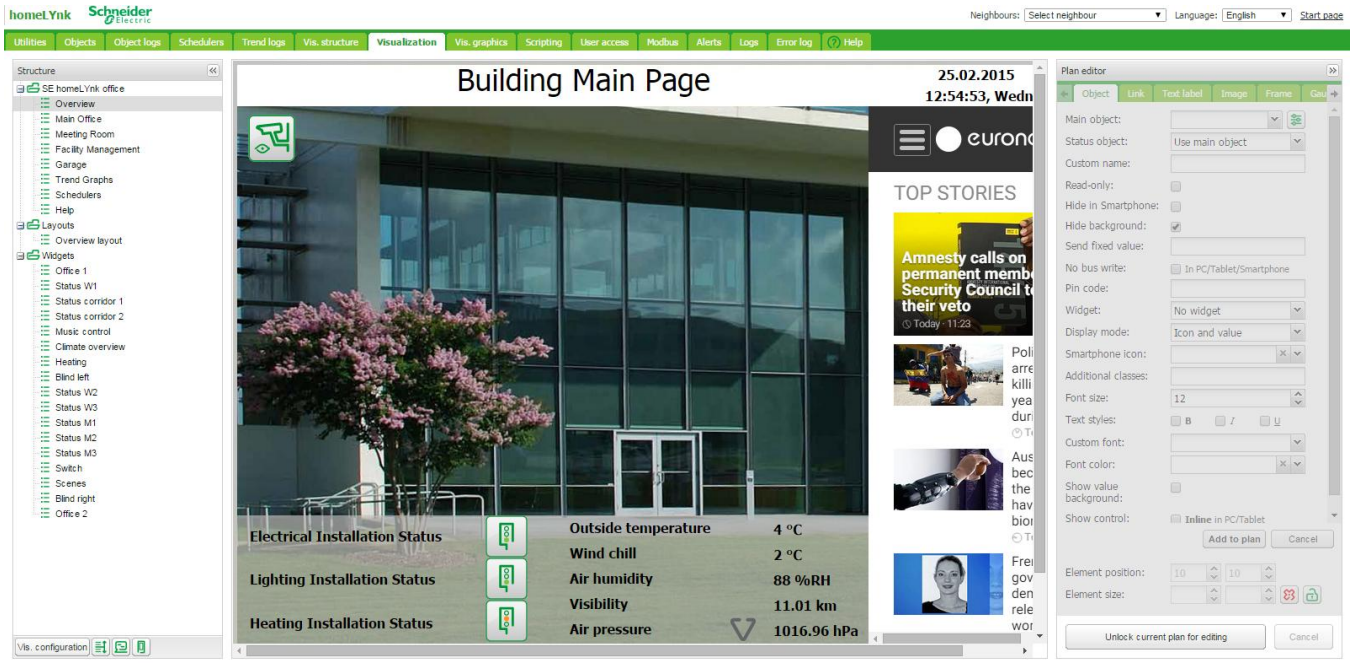
NOTE: Order of objects with the same priority is not defined and it can differ in the editor and PC/Tablet visualization.

2015 Schneider Electric. All rights reserved.

3.9 Visualization

This tab is split into three sections:

1. **Structure** – Navigation tree for levels, plans, widgets which were created under the visualization structure tab.
2. **Visualization map** – Actual visualization field where you can add all visualization components.
3. **Plan Editor** – All parameters of the component are set up here.



Both side bars can be minimized by pressing icon making the plan more visible especially on small displays.

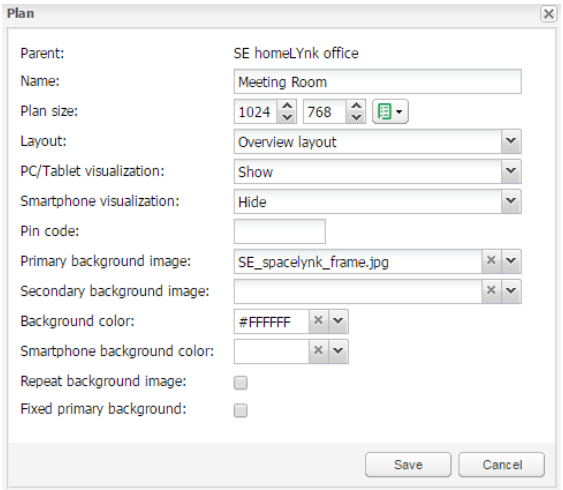
2015 Schneider Electric. All rights reserved.

3.9.1 Structure

To navigate between the plans, layouts and widgets using the navigation tree in the structured view.

In the editing mode the following additional parameters are available:

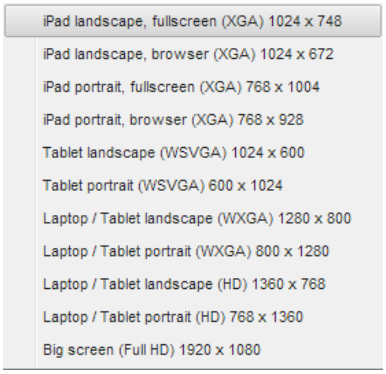
- Size of plans, layouts or widgets.
- Source picture / background color



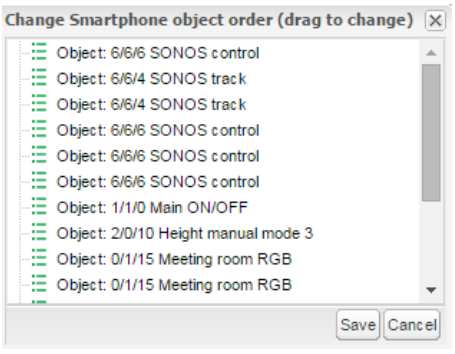
-

NOTE: Size of the plan should be positioned correctly against the background. Widget size has to be always smaller than the plan on which it is placed. Always use the component position to align the objects.

Predefined size of the plan:


iPad landscape, fullscreen (XGA) 1024 x 748
iPad landscape, browser (XGA) 1024 x 672
iPad portrait, fullscreen (XGA) 768 x 1004
iPad portrait, browser (XGA) 768 x 928
Tablet landscape (WSVGA) 1024 x 600
Tablet portrait (WSVGA) 600 x 1024
Laptop / Tablet landscape (WXGA) 1280 x 800
Laptop / Tablet portrait (WXGA) 800 x 1280
Laptop / Tablet landscape (HD) 1360 x 768
Laptop / Tablet portrait (HD) 768 x 1360
Big screen (Full HD) 1920 x 1080

To order the object in the Smartphone visualization, press **Reorder Smartphone objects** button .


Change Smartphone object order (drag to change)
Object: 6/6/6 SONOS control
Object: 6/6/4 SONOS track
Object: 6/6/4 SONOS track
Object: 6/6/6 SONOS control
Object: 6/6/6 SONOS control
Object: 6/6/6 SONOS control
Object: 1/1/0 Main ON/OFF
Object: 2/0/10 Height manual mode 3
Object: 0/1/15 Meeting room RGB
Object: 0/1/15 Meeting room RGB
Save
Cancel

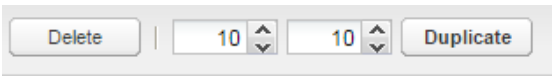
Next to the icon ***Reorder Smartphone objects*** there are two icons for a quick:

- Preview in PC/Tablet.
- Preview in Smartphone.

2015 Schneider Electric. All rights reserved.

3.9.2 Visualization Map

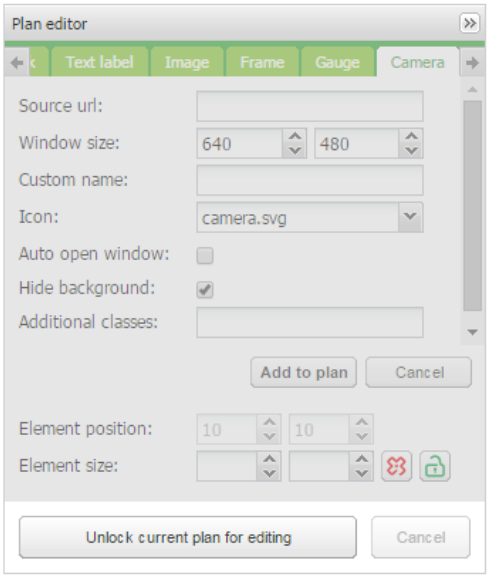
Each newly added object will be placed on the top left corner of the plan with vertical and horizontal spacing predefined in object menu.



Selected object can be resized by pulling strip on the bottom or right side, deleted or duplicated (duplicated object will be displayed with predefined spacing).

3.9.3 Plan editor

Plan editor is located on the right side of the visualization map. Editing mode can be accessed by pressing ***Unlock current plan for editing***.



2015 Schneider Electric. All rights reserved.

3.9.4 Object

Every control or monitoring objects are configured under this tab. Different data types have different parameters.

Object visualization parameters can be changed via pressing icon . It refers to ***Vis.parameter*** in the ***Objects*** tab.

Main object – List of existing group addresses on KNX/EIB bus, the ones available for configuration in the ***Objects*** tab. In order to speed up the selection, it is recommended to start writing group address.

Status object – List of the status objects on KNX/EIB bus. Control object can also be used as status.

Custom name – Name for the object. Custom name is important for Smartphone Visualization; if the name is left blank, the group address name is used instead.

Read-only – The object is read-only, no write (control) permission.

Hide in Smartphone – Do not show this object in the ***Smartphone Visualization***.

Hide background– Hide icon background.

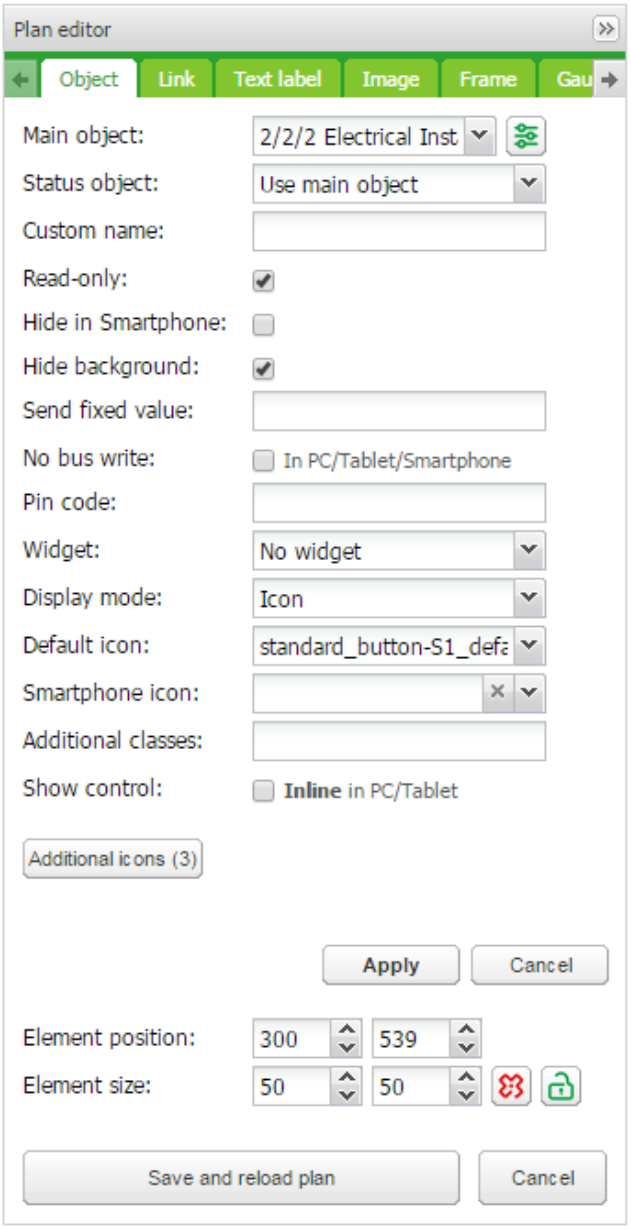
Send fixed value– Allows sending specific value to the bus each time the object is pressed.

No bus write – Value will not be written in to KNX bus. Useful for triggering scripts with bus load limitation.

Pin code – Via adding a pin you can protect the object. Each time the value is changed the pin code will be requested to enter.

Widget –Widget can be attached to a button which needs to be created before. Widget cannot be tested in the editor mode; but only in PC/Tablet Visualization.

Display mode ***[icon and value; icon; value]*** – How to display the object.



2015 Schneider Electric. All rights reserved.

Smartphone Icon– Default icon for Smartphone if differ from PC/tablet one.

On icon – On state icon for binary-type objects.

Off icon – Off state icon for binary-type objects.

Additional classes– Create additional class, which can be used in custom CSS file in order to modify particular group of graphical objects.

Font size– For value display text style can be defined.

Text styles– **[Bold / Italic / Underscore]** option.

Custom font– selection from installed fonts.

Font size– Size of selected font.

Show control – If enabled, any control button graphics will change from a symbol to a switch . Visible only in PC/Tablet Visualization.

For value-type objects, additional button appears while specifying parameters – ***Additional icons***. Different icons for different object values can be defined in the window.

For value display text style can be defined.

After defining the object parameters, press ***Add to plan*** button and a newly created object appears. The object can be moved to any location of the plan.

NOTE: In the editing mode, the object does not work. When all the necessary objects are added, press ***Save and reload floor plan*** button so that the objects start functioning.

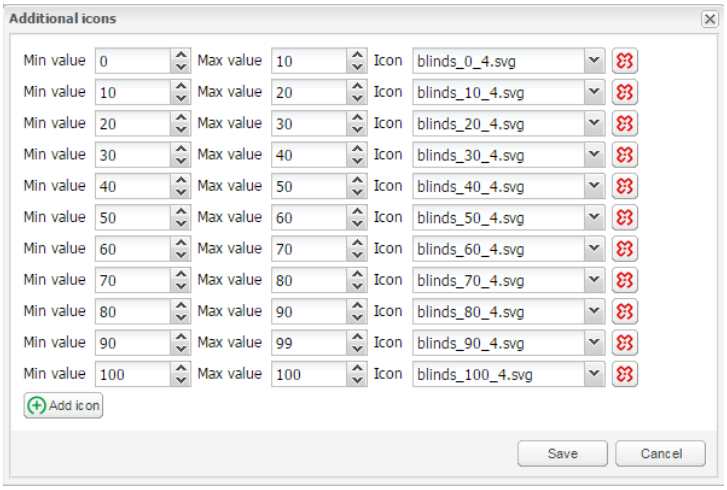
Each added object can be edited while clicking on it in the Editing mode. Press

button after each change.

Each object can be duplicated via pressing

button on the left side.

button will set the object parameters to default settings.


Additional icons
Min value 0 Max value 10 Icon blinds_0_4.svg
Min value 10 Max value 20 Icon blinds_10_4.svg
Min value 20 Max value 30 Icon blinds_20_4.svg
Min value 30 Max value 40 Icon blinds_30_4.svg
Min value 40 Max value 50 Icon blinds_40_4.svg
Min value 50 Max value 60 Icon blinds_50_4.svg
Min value 60 Max value 70 Icon blinds_60_4.svg
Min value 70 Max value 80 Icon blinds_70_4.svg
Min value 80 Max value 90 Icon blinds_80_4.svg
Min value 90 Max value 99 Icon blinds_90_4.svg
Min value 100 Max value 100 Icon blinds_100_4.svg
Add icon
Save
Cancel

2015 Schneider Electric. All rights reserved.

Element position– Can be added manually or by drag and drop of object for X and Y axis position on the plan. ***Element size***– Can be added manually or by dragging vertical horizontal strip o f the object. Object size can be reset to default size by pressing reset icon . Aspect ratio of object can be locked by pressing lock icon .	

2015 Schneider Electric. All rights reserved.

3.9.5 Link

In order to make the visualization more convenient, there are plan links integrated. Special icons on the map can be added which would act as a link to other plans.

Link to – Select plan link.

Custom name –Name for the link.

Hide in Smartphone – Do not show this plan link in *Smartphone Visualization.*

Hide background – Hide the icon background

Display mode ***[icon; value]*** – how to display the plan link.

Icon – Icon which will be shown in the visualization. If only text is selected, text parameters are selected.

Active state icon – If icon is selected, then the active plan icon is available.

Additional classes– Create additional class, which can be used in custom CSS file in order to modify particular group of graphical objects.

Font size – Size of font.

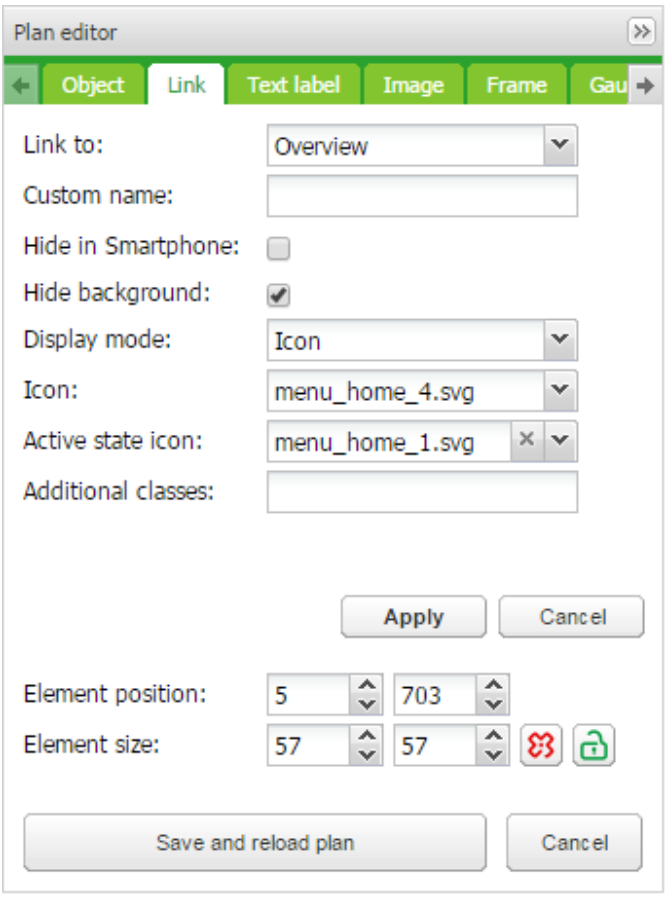
Text style – Text style – bold, italic, underscore.

Custom font – Font name.

Font color – Font color.

Element size and position – see ***Plan editor> Object*** tab.

NOTE: It is recommended to use the Layout for menu and plan link creation. Time can be saved while adding it to different plans and later when making changes. By adding it to different plans it would save time and be beneficial when changes are required.

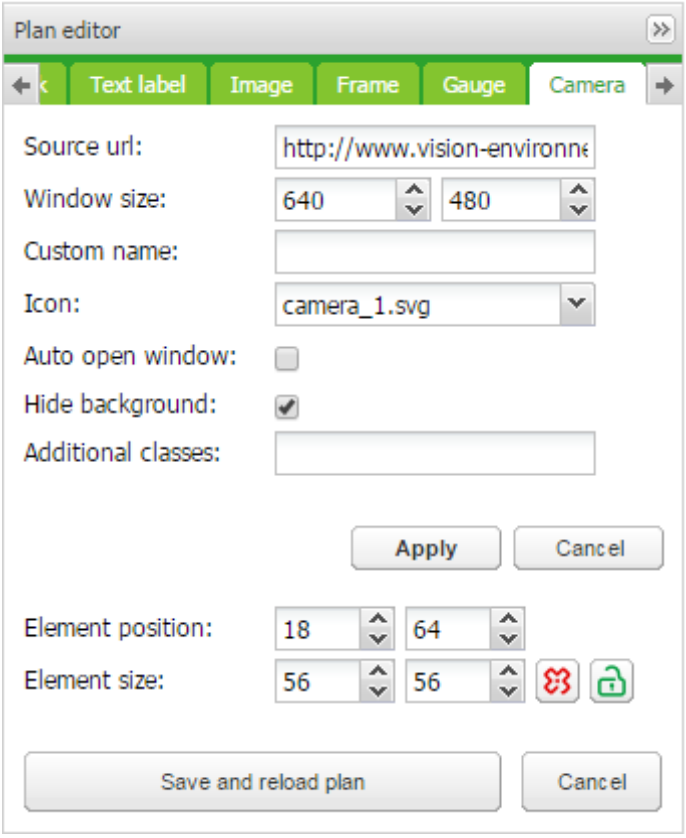


2015 Schneider Electric. All rights reserved.

3.9.6 Camera

homeLYnk supports third party IP web camera integration into its visualization.

NOTE: Only cameras which support HTTP MJPEG streaming in web browser can be visualised.



Source url – Source address of the video stream.

Window size – Width and height for displaying the graph.

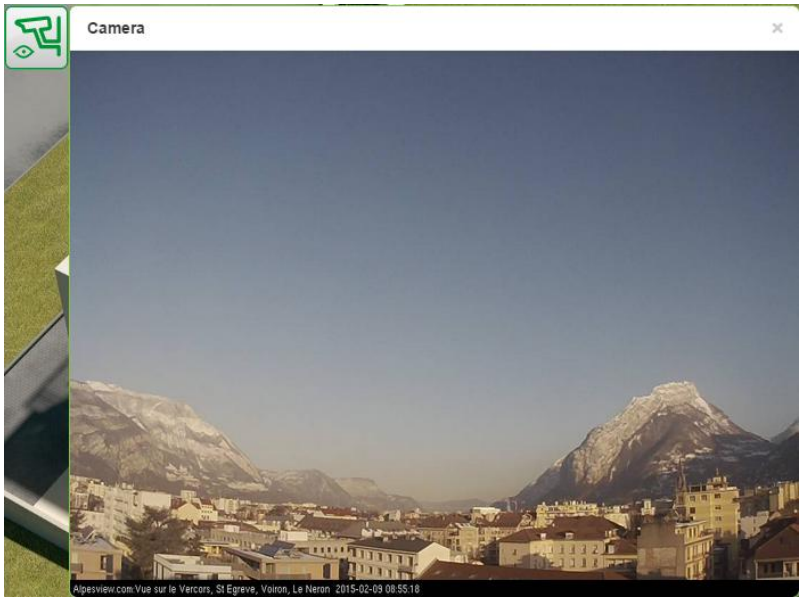
Custom name – Name for the object.

Auto open window – Automatically open video window when plan is open.

Additional classes– Create additional class, which can be used in custom CSS file in order to modify particular group of graphical objects.

Hide background– Hide icon background.

Sort order – Order cameras for touch visualization



NOTE:

- If IP camera requires user name and password, enter the url accordingly: ***http://USER:PASSWORD@IP***
- Feel free to change icon or its label for your camera
- homeLYnk is only a redirecting stream from camera to the browser. If the stream does not work, it is a web browser issue not the homeLYnk.
- If it is a cameras issue, please check if the video stream is available in the browser.
- If the camera is available from external, the IP of the camera need to be port forwarded trough the router. While adding the external camera, IP with the correct port has to be used (**IP:port**). If the local IP is used, then the camera will not be available externally.
- Contact Technical support of the camera manufacturer if the direct video stream is hidden by the manufacturer.

2015 Schneider Electric. All rights reserved.

3.9.7 Graph

Real-time graphs can be integrated into visualization system to monitor the current and the old value of the scale-type objects. Make sure logging is enabled for the object in the ***Object*** tab where values are planned to be shown in the graph.

Data object – Group address of the object.

Object must have ***Log*** option activated for ***Graph*** to be active in ***Visualization***.

Custom name – Name of the object.

Icon– Icon to launch the graph.

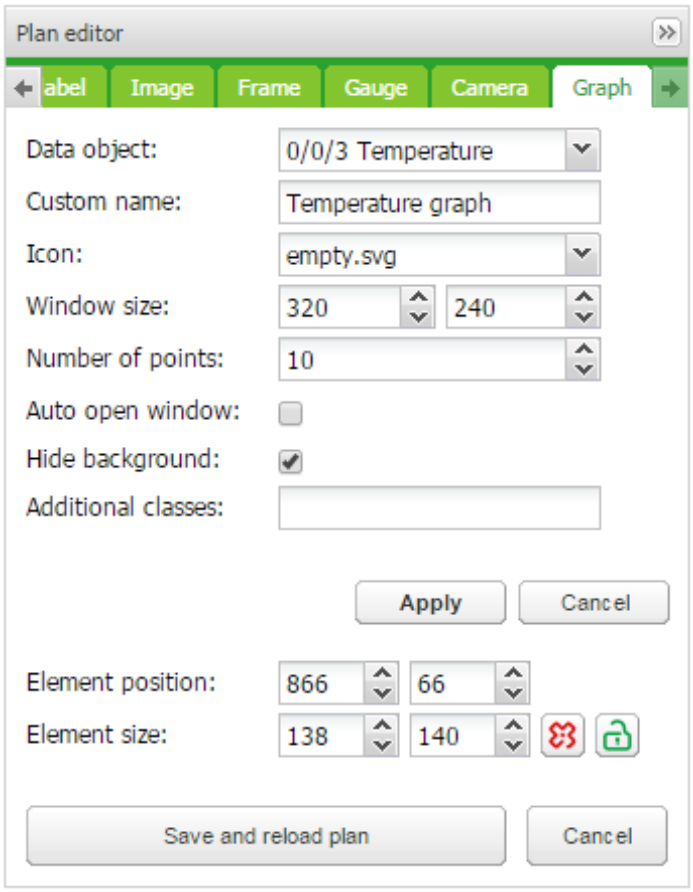
Window size – Width and height for displaying the graph.

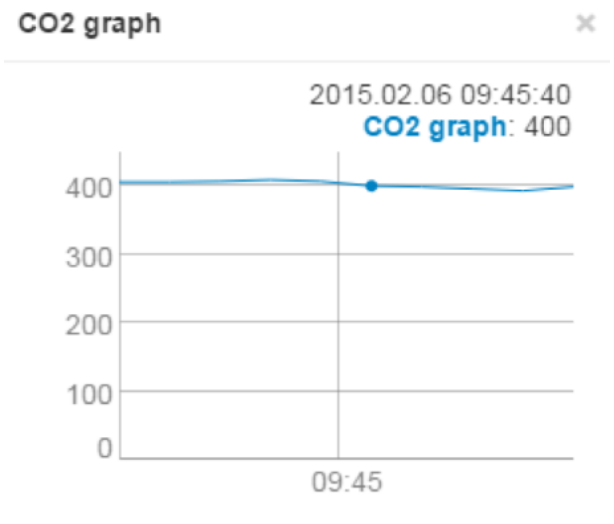
Number of points – Number of data points to show in the graph.

Auto open window – Graph window is automatically opened.

Hide background – Hide icon background.

Additional classes – Create additional class, which can be used in custom CSS file in order to modify particular group of graphical objects.





Once the graph parameters are defined, press ***Add to plan*** button and newly created object will appear. The object can be moved to the desired location.

NOTE: In the editing mode, the graph will not work. Press ***Save and reload plan*** button so that the objects start functioning. (With delay for obtaining relevant data.) Object from which the data are obtained must be set as logged in **Object's properties**.

2015 Schneider Electric. All rights reserved.

3.9.8 Text label

Text labels can be added and moved across the visualization map.

Text – Label text

Font size – Label font size

Text style – Style of the text – bold, italic, underscored.

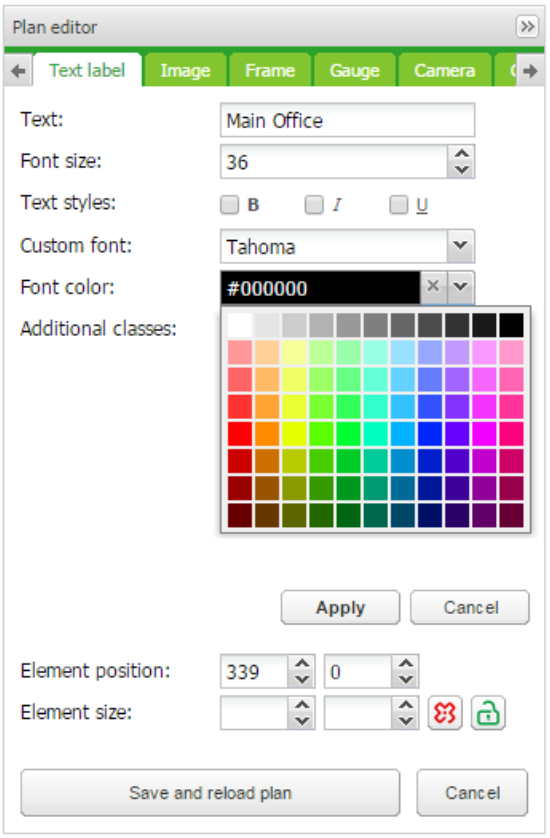
Custom font – Font name.

Font color– Label font color-

Additional classes – For custom CSS styles.

Once the label parameters are defined, press ***Add new object*** button and newly created label will appear on the map. The object can be moved to the desired location. Press on ***Save and reload floor plan*** button so the label starts functioning.

Last two rows in the color palette refer to the predefined Schneider Electric corporate colors.



3.9.9 Image

Image section allows adding images from Local storage or from the internet into the visualization map. External image is useful for example, to grab dynamic weathercast images.

Image source ***[Local, Remote]*** – Select image source.

Select image – Select image previously added to ***Vis. graphics -> Images/Backgrounds*****.**

Image url – Source URL of the image.

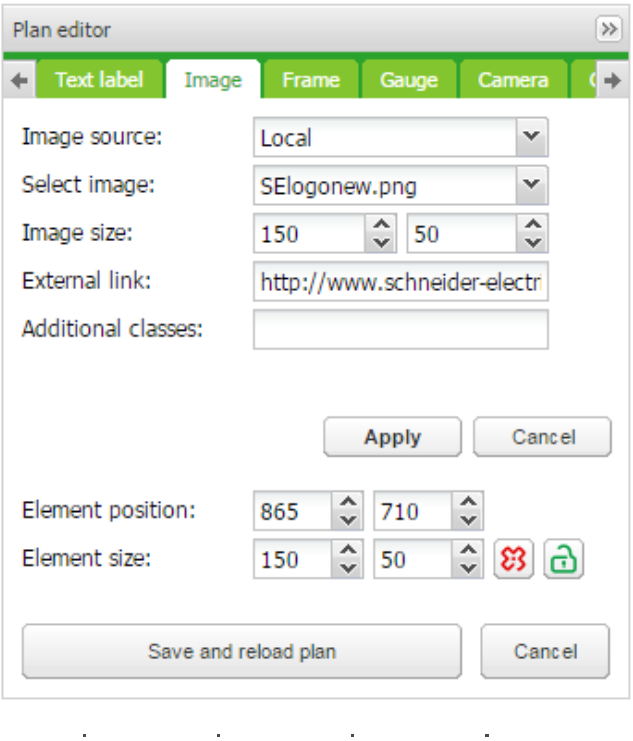
Width – Width of the image.

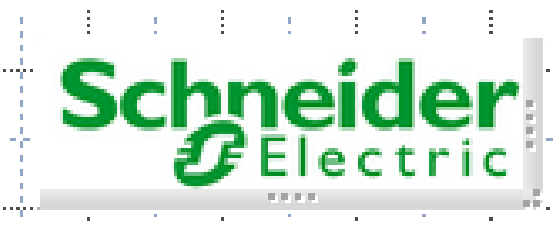
Height – Height of the image.

External link – External link URL when pressing the image example: **http://www.schneider-electric.com/**

Additional classes – For custom CSS styles.

Once the image parameters are defined, press ***Add to plan*** button and newly created object will appear on the map. The object can be moved to the desired location. Image can be freely resized via holding the edge of the image and move. Press ***Save and reload plan*** to apply changes.





2015 Schneider Electric. All rights reserved.

3.9.10 Gauge

Gauge allows dynamic way of visualization and changing the object value in the gauge.

Data object – KNX group address.

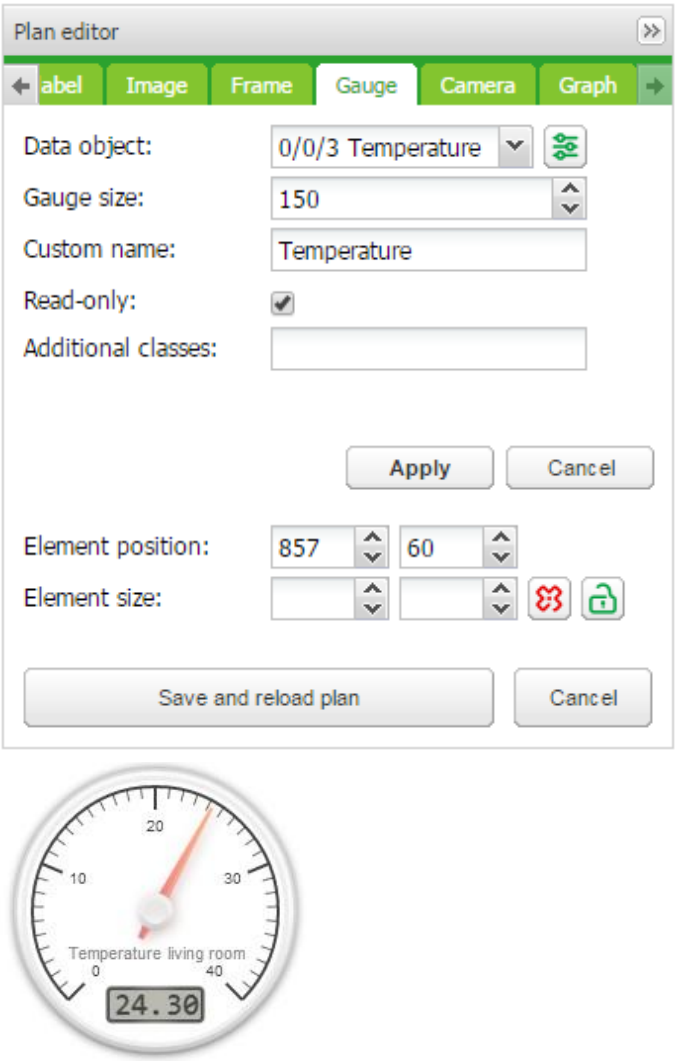
Gauge size – Size of the gauge.

Custom name – Custom name for the object.

Read only – Make the gauge read only.

Additional classes – Create additional class, which can be used in custom CSS file in order to modify particular group of graphical objects.

After defining the gauge parameters press ***Add to plan*** button and newly created object will appear on the map. The object can be moved to the desired location. Press ***Save and reload plan*** button so that the gauge starts functioning.



2015 Schneider Electric. All rights reserved.

3.9.11 Frame

Frame allows displaying internal or external webpage in visualization. ***Schedulers*** and ***Trends*** can be integrated into the frame.

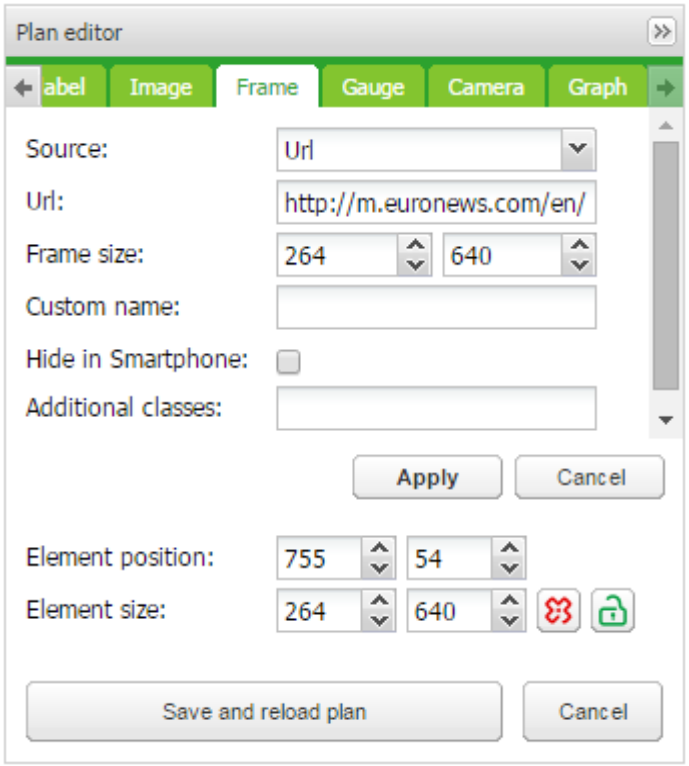
Source – Select Scheduler, Trend log or external URL.

Url: - Source URL of external webpage.

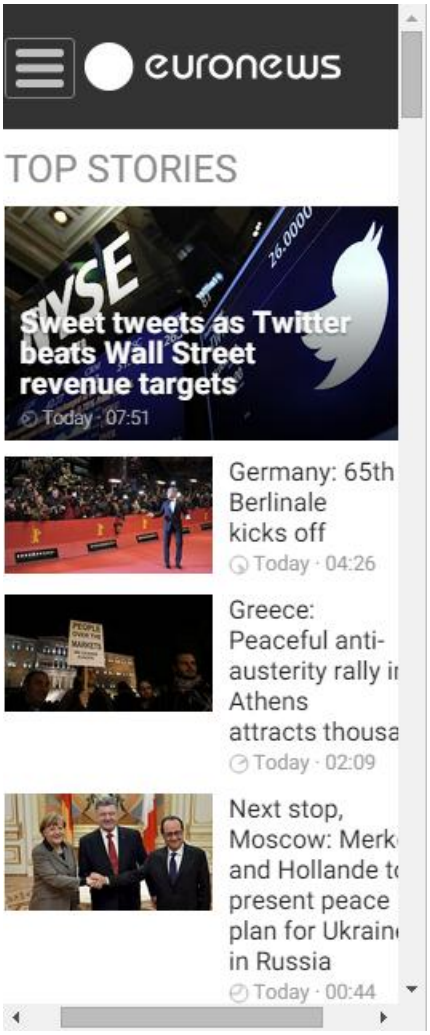
Frame Size: Width/Height of the frame

Custom name - Specify the title of the frame.

Hide in Smartphone - When ticked, not available in the Smartphone visualization.



After defining the frame parameters, press ***Add to plan*** button and newly created object will appear on the map. The frame can be moved to the desired location. Frame can be freely resized via holding the edge of the Frame and move. Press ***Save and reload plan*** button so the frame starts functioning.


euronews
TOP STORIES
Sweet tweets as Twitter beats Wall Street revenue targets
Today · 07:51
Germany: 65th Berlinale kicks off
Today · 04:26
Greece: Peaceful anti-austerity rally i Athens attracts thousa
Today · 02:09
Next stop, Moscow: Merk and Hollande t present peace plan for Ukrain in Russia
Today · 00:44

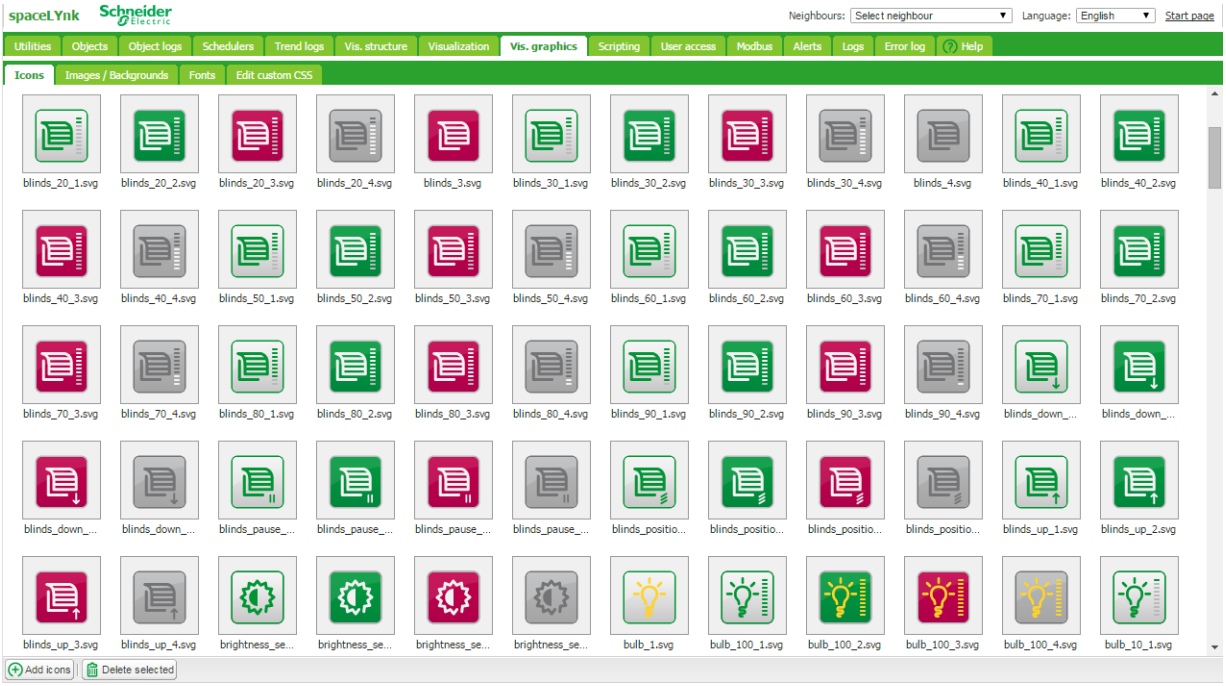
NOTE:

- Some web pages have java script which prevent from using frame, if this is implemented, the webpage will open in full screen rather in the frame
- It is recommended to stretch the frame to maximum width if Scheduler or Trend is used. Recommended minimum width is 1024.
- Frame is only visible under PC/Tablet Visualization.
- Do not allow Scheduler or Trend to be viewed from Smartphone visualization. Settings are available in ***Vis. structure*** under dedicated plan.

2015 Schneider Electric. All rights reserved.

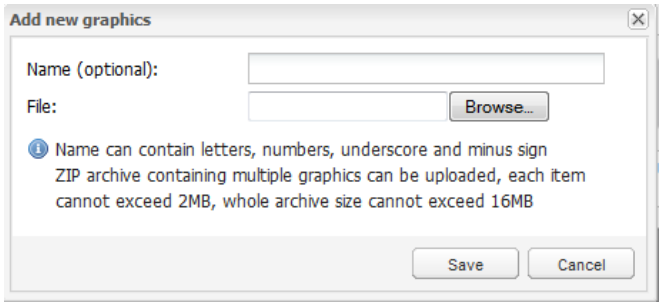
3.10 Vis. graphics

This tab is split into three sections. ***Icons*** where all object icons are located, ***Images/Backgrounds*** for all the locally stored pictures and ***Edit custom CSS*** to create or edit the custom cascade style sheets.



Press ***Add new icon*** button to add a new entry. The system accepts any icon size.

Jpeg, Gif, PNG and SVG formats are supported. Name can contain letters, numbers, underscore and minus sign.
ZIP archive containing multiple graphics can be uploaded, each item cannot exceed 2MB, and whole archive size cannot exceed 16MB.


Add new graphics
Name (optional):
File:
Browse...
Name can contain letters, numbers, underscore and minus sign
ZIP archive containing multiple graphics can be uploaded, each item cannot exceed 2MB, whole archive size cannot exceed 16MB
Save
Cancel

Name (optional) – The name of the icon. It will appear in the list when adding new object. It can contain letters, numbers, underscore and minus sign.

File – Icon file location.

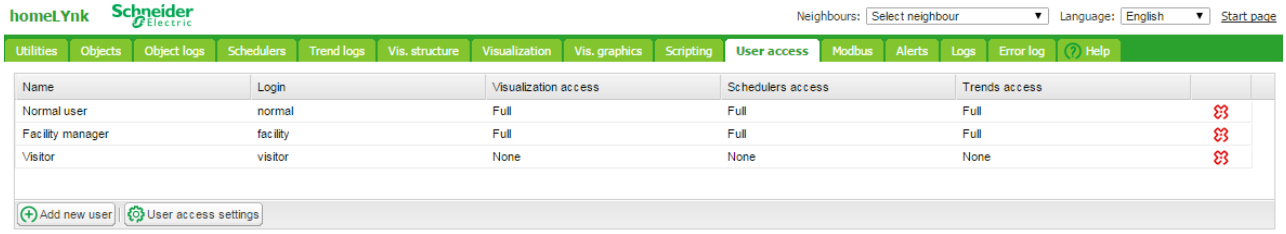
CSS style can be changed via uploading new file. CSS define all control buttons, Smartphone visualization, Scheduler and Trend. For more information on how to modify the CSS file, please contact your local front office for additional document.

NOTE: Clear cache of the browser after uploading new CSS file.

2015 Schneider Electric. All rights reserved.

3.11 User access

User access menu allows creating and managing user's accounts.



NOTE: Amount of users is limited to 8 in homeLYnk.

2015 Schneider Electric. All rights reserved.

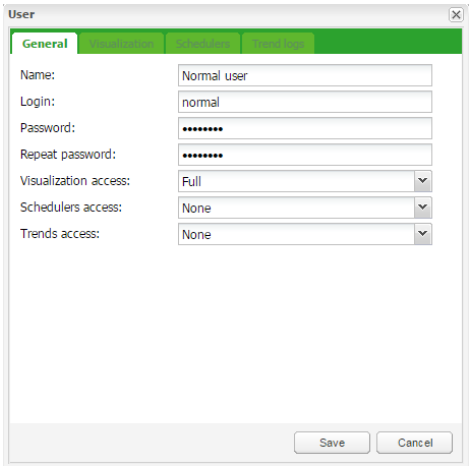
Adding new user

Click on Add new user icon to add new user.

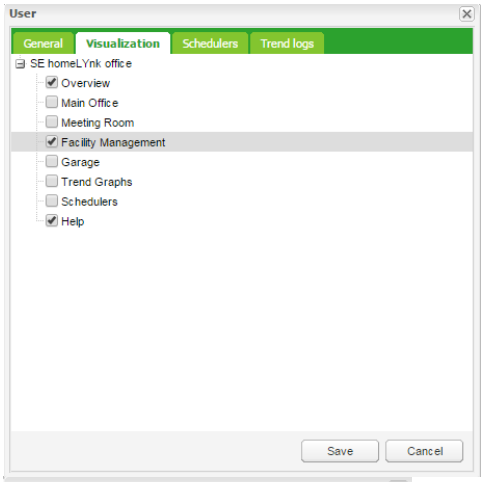
User name – Name of account.

Login – Length 2 to 20 characters, accepted characters: "-", "_", "a-z", "A-Z", "0-9".

Password – Length 6 to 20 characters. Any character accepted.

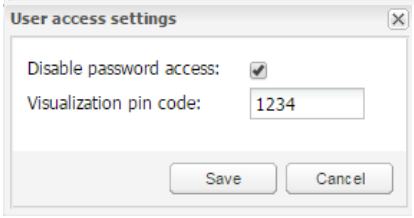


Visualization/Schedulers/Trend access – [None/Partial/Full]. When Partial access selected, particular Visualization plans/Schedulers/Trend logs can be selected.



User access settings

Click on User access settings icon to manage user access settings.



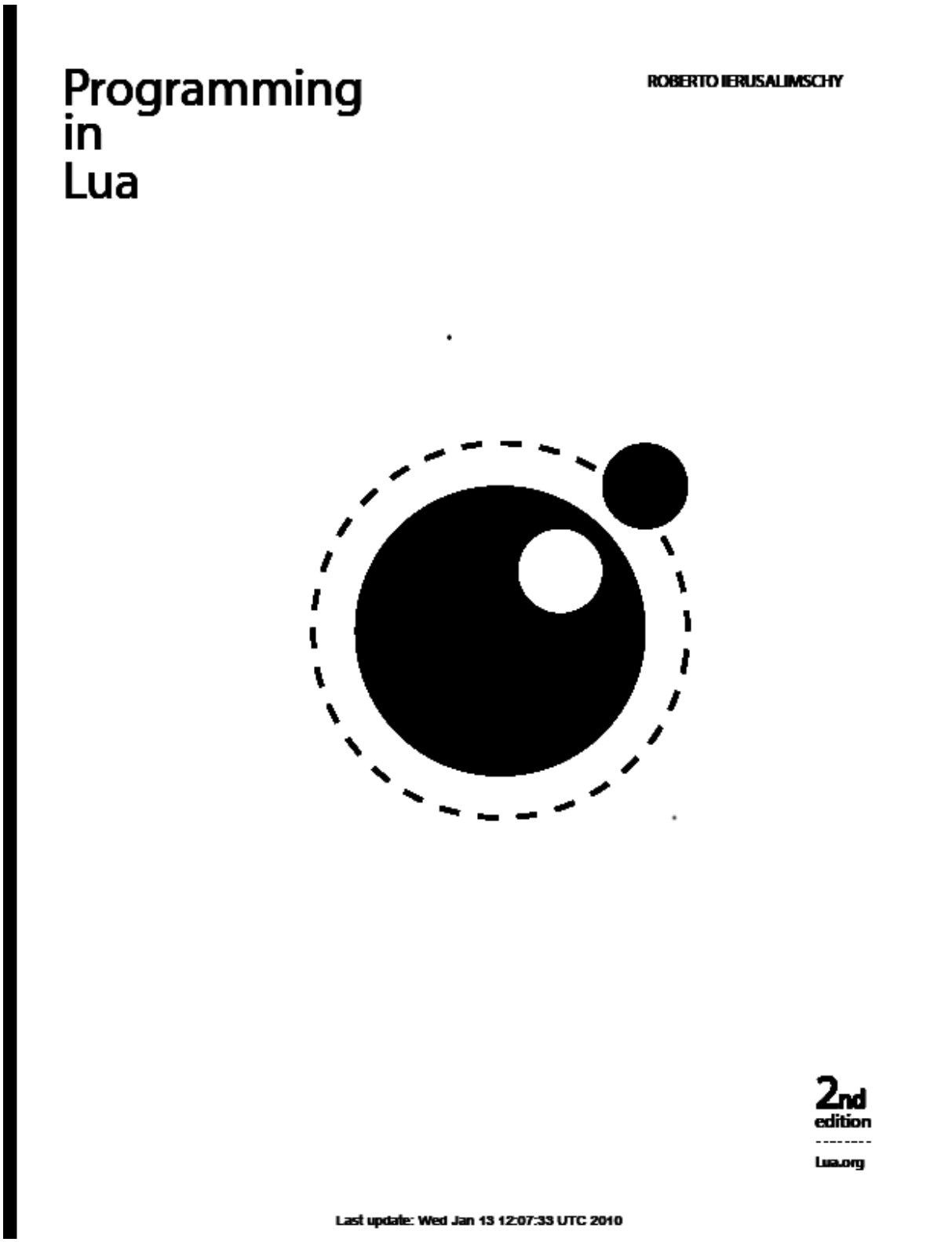
Disable password access – When active password protection is disabled.

Visualization pin code –When active password is disabled access can be protected by common Pin code. Length 3 to 8 characters numbers only.

2015 Schneider Electric. All rights reserved.

3.12 Scripting

Scripting menu allows adding and managing various scripts, depending on the type of the script. Lua programming language is used to implement user scripts. Most of the Lua language aspects are covered in the first edition of ***Programming in Lua*** which is freely available at http://lua.org/pil/.



2015 Schneider Electric. All rights reserved.

3.12.1 Event based

NOTE: Data format — in most cases data is stored and transferred between homeLYnk parts using hex-encoded strings (2 bytes per 1 byte of data).

These are scripts that are executed when a group event occurs on the bus. Usually used when real-time response is required.

When pressing the arrow on the lower side of the ***Event-based***, ***Resident*** or ***Scheduled*** buttons, two options appear:

List view – Sort scripts in list view.

Add new script – Add new script to the list

The following fields should be filled when adding a new script:

Script name – The name of the script.

Group address/tag – Group addresses must be entered. Tags are listed in the drop-down.

Group address - Allows to enter only digits from 0.9 and / as a separator. When icon appears on the right side of the text-box, wrong address form is used. Correct form of the group-address is, for example, 1/1/1.

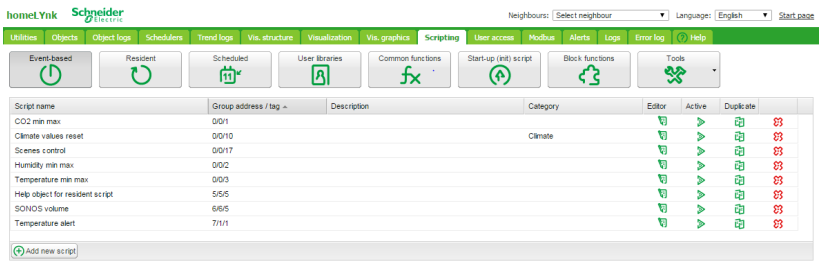
Tag – Script can run on tags. If group addresses have tag attached to and script is using tag, then any telegram which is sent to the group with this tag will execute the script.

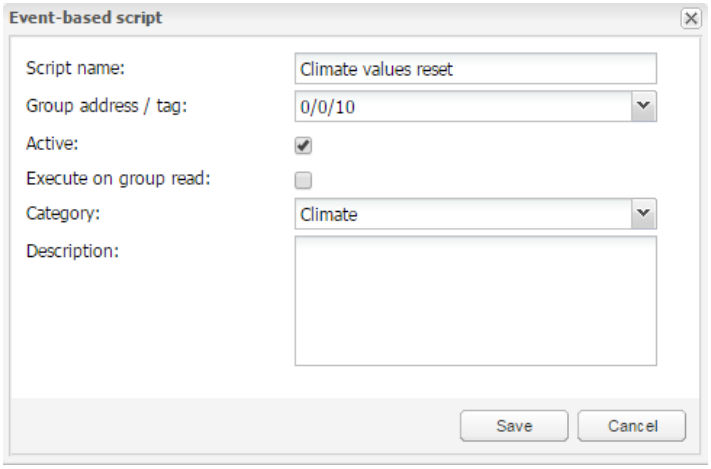
Active– Specifies whether the script is active (green circle) or disabled (red circle).

Execute on group read– Specifies whether the script is executed on KNX group read telegram.

Category – A new or existing name of the category the script will be included. This will not affect on script action, helps only by grouping the scripts and watching by categories in ***Tools*** ***Print*** script listings page.

Description– Description of the script.





NOTE: If the script is run only on read request, use the following script example:

```
if event.type == 'groupread' then

-- script here

end
```


2015 Schneider Electric. All rights reserved.

3.12.2 Resident

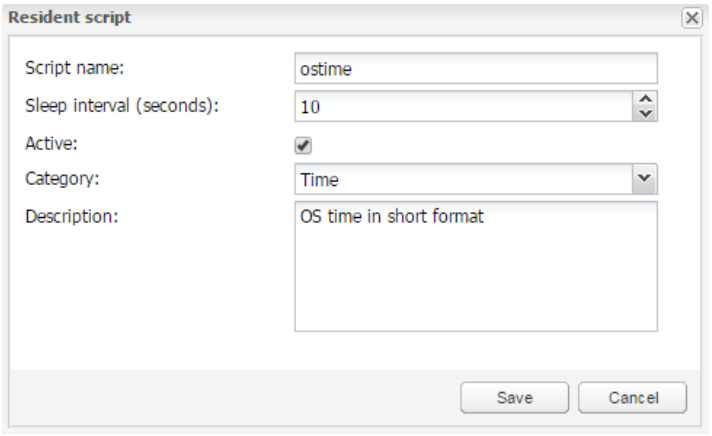
Script name – The name of the script

Sleep interval (seconds) – Interval after which the script will be executed.

Active– Specifies whether the script is active (green circle) or disabled (red circle).

Category – A new or existing name of the category the script will be included. This will not affect on script action, helps only by grouping the scripts and watching by categories in **Tools Print script** listings page.

Description– Description of the script



3.12.3 Scheduled

Script name – The name of the script.

Minute – Minute.

Hour – Hour.

Day of the month – Day of the month.

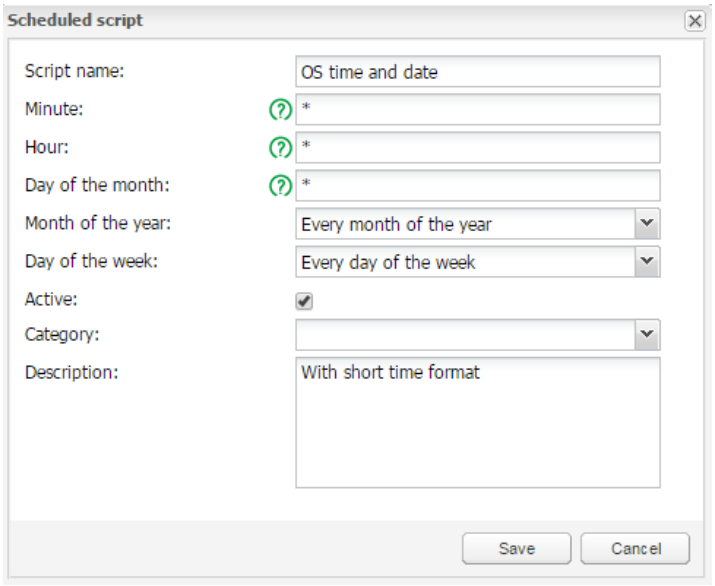
Month of the year – Month of the year.

Day of the week – Day of the week.

Active– Specifies whether the script is active (green circle) or disabled (red circle).

Category – A new or existing name of the category the script will be included. This will not affect on script action, helps only by grouping the scripts and watching by categories in ***Tools Print*** script listings page.

Description– Description of the script.



2015 Schneider Electric. All rights reserved.

3.12.4 User libraries

User libraries usually contain user defined functions which are later called from other scripts.

Secure the Code

There is an option ***Keep source*** available for user libraries. Once disabled, the code is compiled in the binary form and cannot be seen for further editing. If this option is enabled, the source code is seen in the editor.

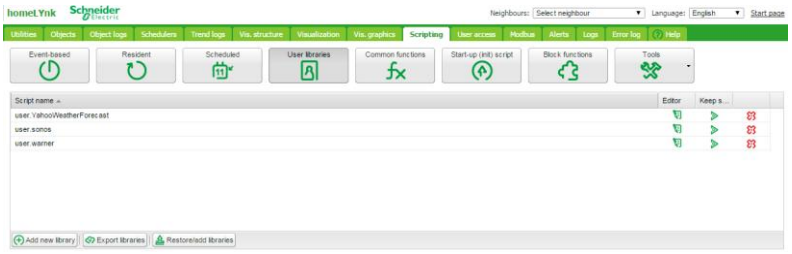
Auto load library option will load selected script when homeLYnk starts.

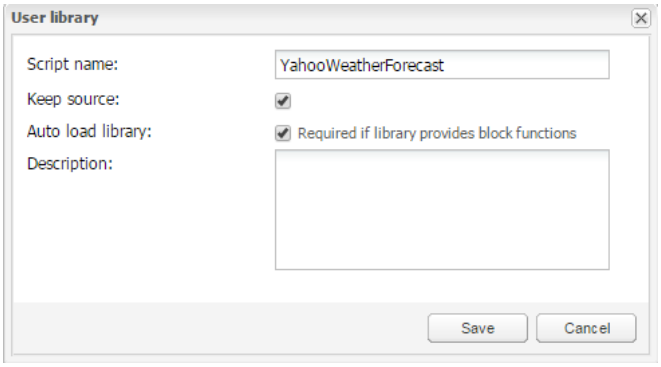
Include the Library in the Scripts

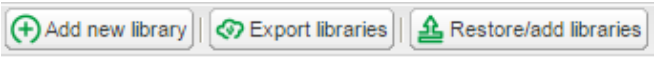
To use functions defined in user library, they should be included in the beginning of the script, for example, user library with the name 'test' should be included as below :

require('user.test')

User Libraries can be backed up and restored/added from archive.

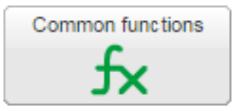


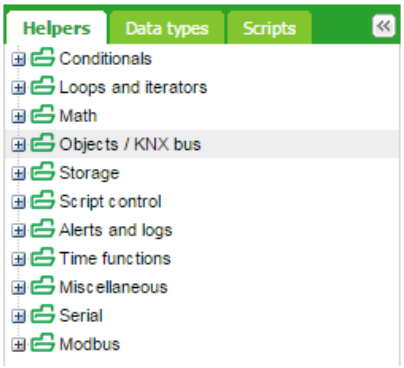




3.12.5 Common functions

Common functions contain library of globally used functions. They can be called from any script, any time, without special inclusions with ***User libraries***. Functions like sunrise/sunset; Email is included by default in ***Common functions***.





3.12.6 Start-up script

Init script is used for initialization on specific system or bus values on system start. Init script is run each time after the system has restarted (power up, reboot in the SW or via hardware **RESET** push button).

2015 Schneider Electric. All rights reserved.

3.12.7 Tools

Export helpers – Export scripting helpers, save with the right mouse click.

Import helpers – Import scripting helpers.

Restore helpers – Restore default scripting helpers.

Backup user scripts – Backup all scripts in *.gz file.

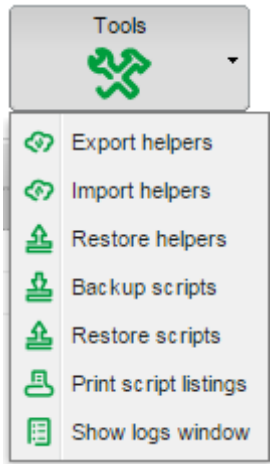
NOTE: Script backup does not backup user libraries, those have to be backed up separately.

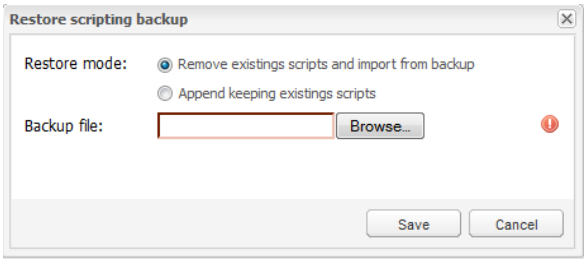
Restore scripts – Restore script from archive (*.gz) file with two possibilities:

- Remove existing scripts and import from backup.
- Append keeping existing (s) scripts.

Print script listings – Shows all scripts with codes in list format sorted by Categories.

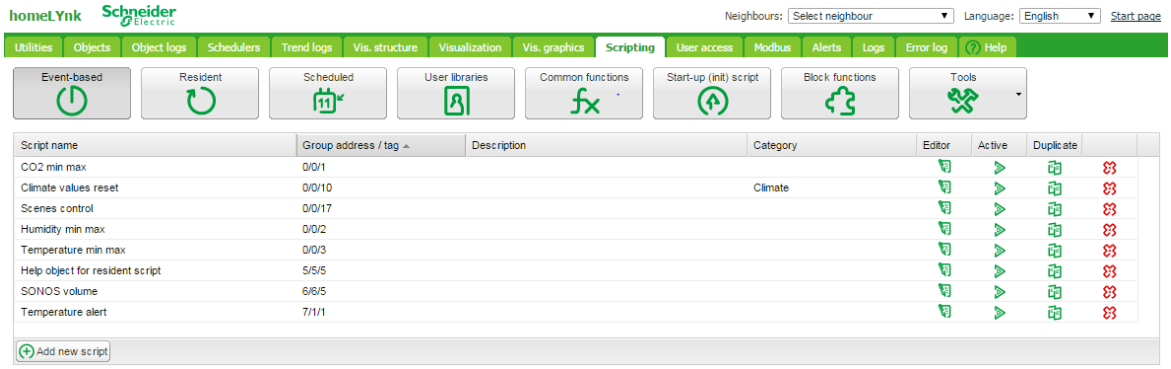
Show logs window – All log data are listed here; it is a duplicated window ***Configuration/Logs***. It allows debugging a script and in parallel checking the logged data.





2015 Schneider Electric. All rights reserved.

3.12.8 General scripting description



There are five actions you can do with each script:

- ***Duplicate*** – Duplicate the script with its source code.
- ***Editor*** – Enter scripting editor to write specific code for the particular program.
- ***Active*** – Make script active (green) or deactivate it (red).
- ***Edit*** – Edit script name, description, category and other parameters.
- ***Delete*** – Delete the script. When pressing this icon the confirmation is asked to accept the delete.

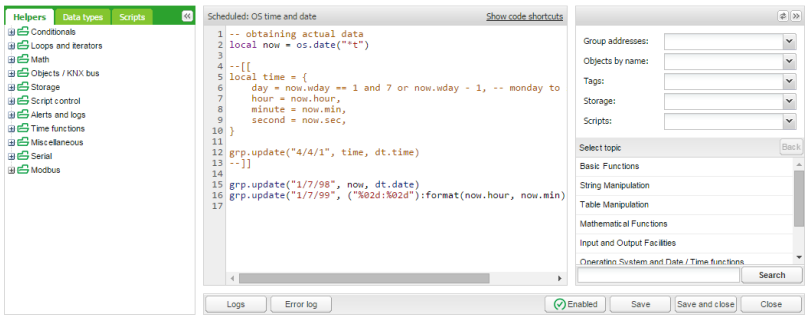
2015 Schneider Electric. All rights reserved.

3.12.9 Script Editor

When a script is added icon appears in the *Editor* column that allows opening a script in the scripting editor and re-working it with built-in code snippets. Code snippets save time and make the coding convenient. After clicking on the appropriate snippet, it automatically adds code to the editor field.

Keyboard shortcuts are implemented for help with script writing.

Logs and error window are also available.



Ctrl + F – Find syntax in a code, text will be highlighted in yellow.

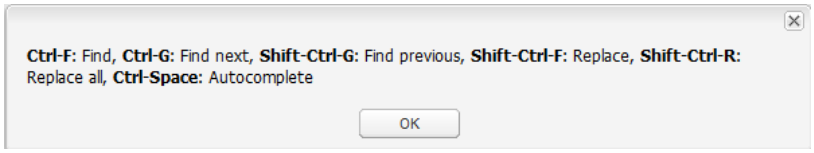
Ctrl + G – After finding a text via Ctrl+F, we can use Ctrl +G to select the next syntax in a script.

Shift + Ctrl + G – Select previous syntax.

Shift + Ctrl + F – Replace syntax in a script by another one. You will be allowed to choose one by one if you want to change it.

Shift + Ctrl + R - Replace all syntaxes in a script by another one at once.

Ctrl + Space – Helps to auto detect code and write for you. Press Ctrl + Space and write first letter of a command, then select the correct one from the list.


Ctrl-F: Find, Ctrl-G: Find next, Shift-Ctrl-G: Find previous, Shift-Ctrl-F: Replace, Shift-Ctrl-R: Replace all, Ctrl-Space: Autocomplete
OK

There are six main groups of Script editor:

Helpers – Predefined code snippets, like if-then statement. Helpers consist of eleven main sub-groups:
Conditionals – If Else If, If Then etc.

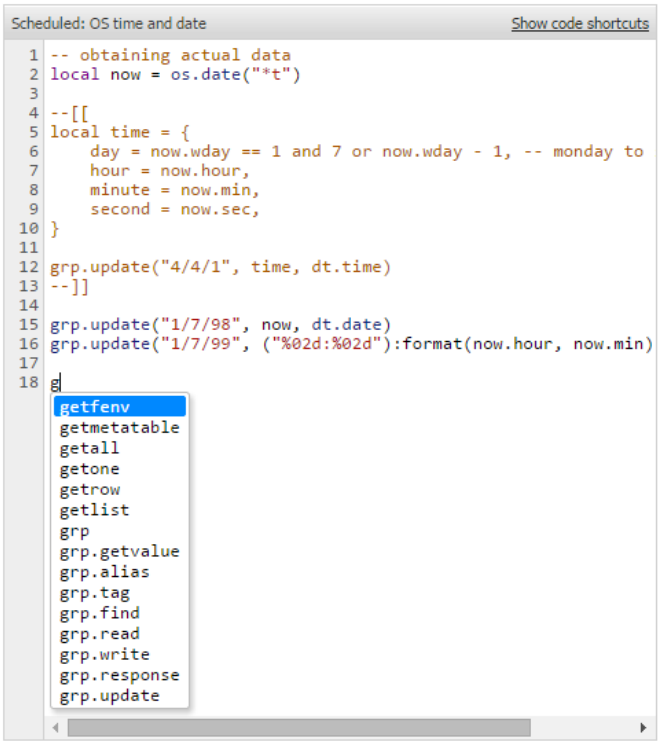
Loops and iterators – Array, Repeat...Until etc.

Math – Random value, Ceiling, Absolute value, Round etc.

Objects/KNX bus – Get object value, Group read, Group write, Update interval etc.

Storage – Get data from storage, Save data to storage.

Script control – Get other script status, enable or disable other scripts.



2015 Schneider Electric. All rights reserved.

Alerts and logs – Alert, Log variables, Formatted alert.

Time functions – Delay script execution.

Miscellaneous – Sunrise/sunset etc.

Serial – Communication through internal homeLYnk I/O ports.

Modbus – Create RTU/TCP connection, Write register, Read register etc.

Group addresses – Existing group addresses on the KNX bus.

Objects by name – Chose object by name.

Tags – Choose object by tag.

Data types – Choose object by data type.

Scripts – List of already made scripts consist 4 sub-groups:

Event –based- List of event-based scripts.

Resident- List of resident scripts.

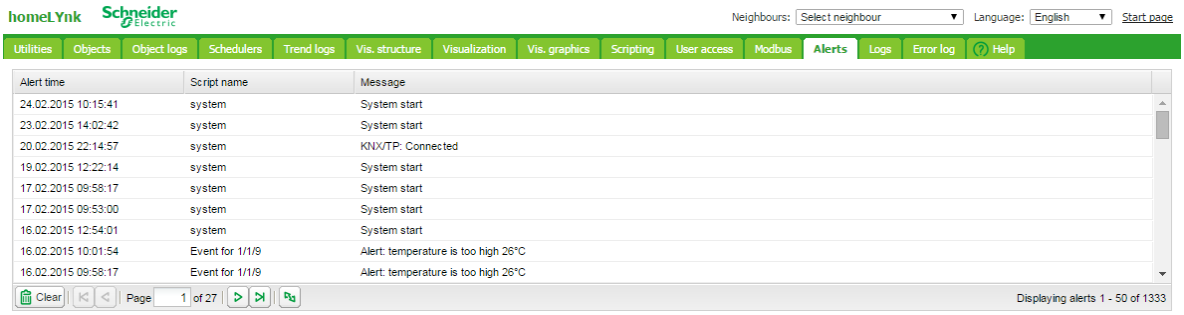
Scheduled – List of scheduled scripts.

User libraries - List of common functions a Star-up (init) scripts.

2015 Schneider Electric. All rights reserved.

3.13 Alerts

In the ***Alerts*** tab a list of alert messages defined with alert function in scripts are located. The messages are stored in the main database.



alert(message, [var1, [var2, [var3]]])

Stores alert message and current system time in the main database.

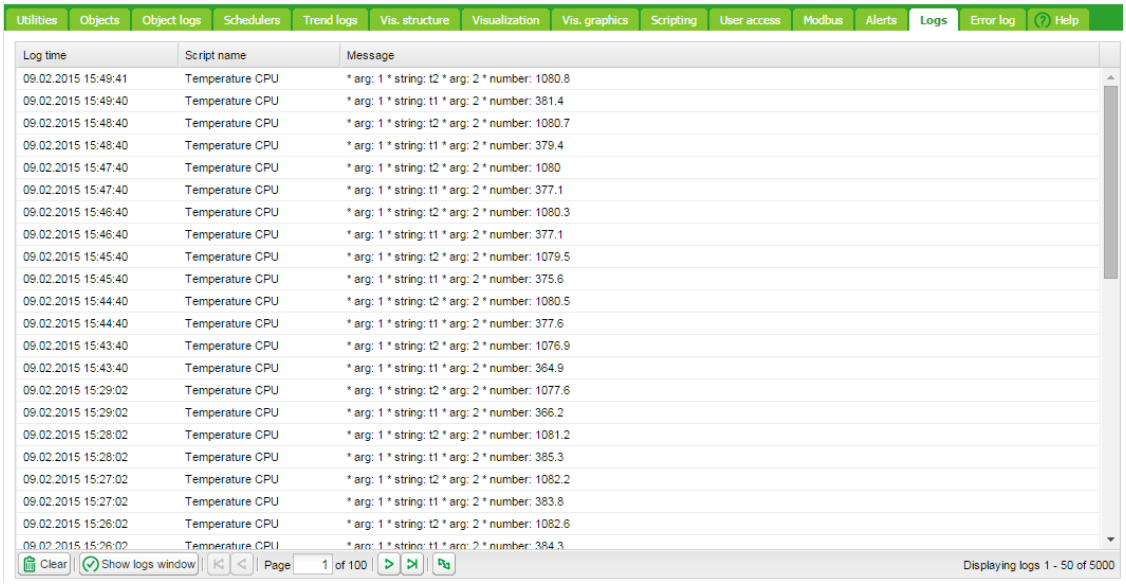
Example:

```
temperature = 25.3

 if temperature > 24 then
-- resulting message: 'Temperature levels are too high: 25.3'

alert('Temperature level is too high: %.1f', temperature)

end
```


2015 Schneider Electric. All rights reserved.

3.14 Logs

Logs can be used for scripting code debugging. The log messages appear defined by *log* function.


Utilities
Objects
Object logs
Schedulers
Trend logs
Vis. structure
Visualization
Vis. graphics
Scripting
User access
Modbus
Alerts
Logs
Error log
Help
Log time
Script name
Message
09.02.2015 15:49:41 Temperature CPU * arg: 1 * string: t2 * arg: 2 * number: 1080.8
09.02.2015 15:49:40 Temperature CPU * arg: 1 * string: t1 * arg: 2 * number: 381.4
09.02.2015 15:48:40 Temperature CPU * arg: 1 * string: t2 * arg: 2 * number: 1080.7
09.02.2015 15:48:40 Temperature CPU * arg: 1 * string: t1 * arg: 2 * number: 379.4
09.02.2015 15:47:40 Temperature CPU * arg: 1 * string: t2 * arg: 2 * number: 1080
09.02.2015 15:47:40 Temperature CPU * arg: 1 * string: t1 * arg: 2 * number: 377.1
09.02.2015 15:46:40 Temperature CPU * arg: 1 * string: t2 * arg: 2 * number: 1080.3
09.02.2015 15:46:40 Temperature CPU * arg: 1 * string: t1 * arg: 2 * number: 377.1
09.02.2015 15:45:40 Temperature CPU * arg: 1 * string: t2 * arg: 2 * number: 1079.5
09.02.2015 15:45:40 Temperature CPU * arg: 1 * string: t1 * arg: 2 * number: 375.6
09.02.2015 15:44:40 Temperature CPU * arg: 1 * string: t2 * arg: 2 * number: 1080.5
09.02.2015 15:44:40 Temperature CPU * arg: 1 * string: t1 * arg: 2 * number: 377.6
09.02.2015 15:43:40 Temperature CPU * arg: 1 * string: t2 * arg: 2 * number: 1076.9
09.02.2015 15:43:40 Temperature CPU * arg: 1 * string: t1 * arg: 2 * number: 364.9
09.02.2015 15:29:02 Temperature CPU * arg: 1 * string: t2 * arg: 2 * number: 1077.6
09.02.2015 15:29:02 Temperature CPU * arg: 1 * string: t1 * arg: 2 * number: 366.2
09.02.2015 15:28:02 Temperature CPU * arg: 1 * string: t2 * arg: 2 * number: 1081.2
09.02.2015 15:28:02 Temperature CPU * arg: 1 * string: t1 * arg: 2 * number: 385.3
09.02.2015 15:27:02 Temperature CPU * arg: 1 * string: t2 * arg: 2 * number: 1082.2
09.02.2015 15:27:02 Temperature CPU * arg: 1 * string: t1 * arg: 2 * number: 383.8
09.02.2015 15:26:02 Temperature CPU * arg: 1 * string: t2 * arg: 2 * number: 1082.6
Clear
Show logs window
Page 1 of 100
Displaying logs 1 - 50 of 5000

log(var1, [var2, [var3, ...]])

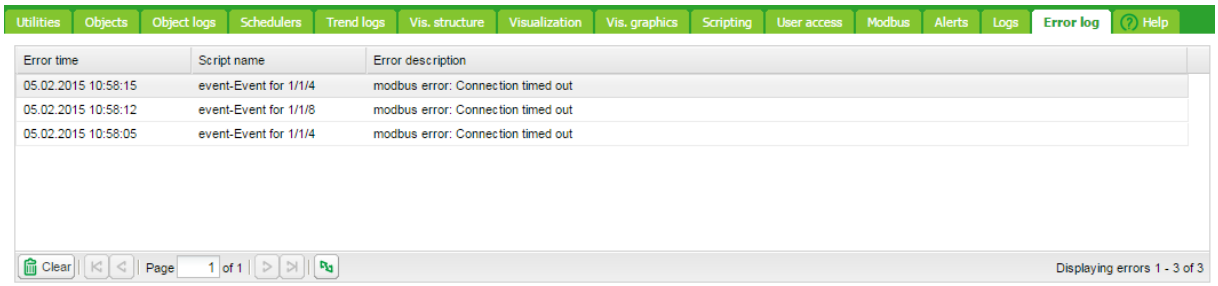
Converts variables to human-readable form and stores them.

Example:

```
-- log function accepts Lua nil, boolean, number and table (up to 5 nested levels) type variables

a ={ key1 ='value1', key2 =2}

b ='test'

c =123.45

-- logs all passed variables

log(a, b, c)
```

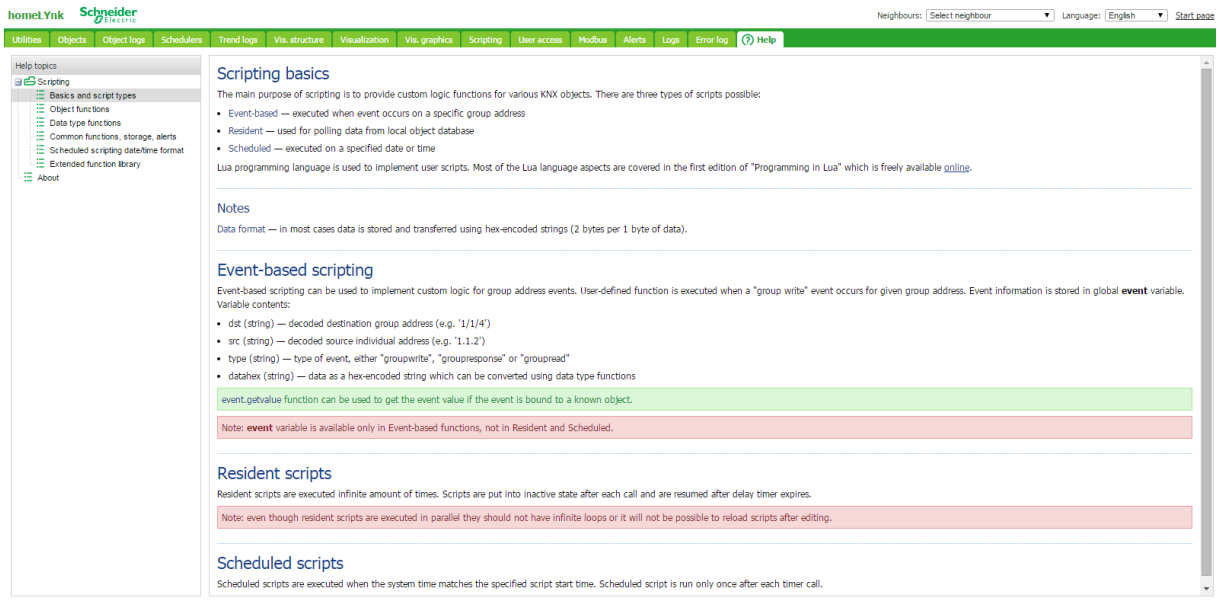

2015 Schneider Electric. All rights reserved.

3.15 Error Log



Error messages from scripts are displayed in Error log tab.

3.16 Help


homeLYnk
Schneider Electric
Neighbours: Select neighbour
Language: English
Start page
Utilities
Objects
Object logs
Schedulers
Trend logs
Vis. structure
Visualization
Vis. graphics
Scripting
User access
Modbus
Alerts
Logs
Error log
Help
Help topics
Scripting
Basics and script types
Object functions
Data type functions
Common functions, storage, alerts
Scheduled scripting date/time format
Extended function library
About
Scripting basics
The main purpose of scripting is to provide custom logic functions for various KNX objects. There are three types of scripts possible:
Event-based — executed when event occurs on a specific group address
Resident — used for polling data from local object database
Scheduled — executed on a specified date or time
Lua programming language is used to implement user scripts. Most of the Lua language aspects are covered in the first edition of "Programming in Lua" which is freely available online.
Notes
Data format — in most cases data is stored and transferred using hex-encoded strings (2 bytes per 1 byte of data).
Event-based scripting
Event-based scripting can be used to implement custom logic for group address events. User-defined function is executed when a "group write" event occurs for given group address. Event information is stored in global event variable. Variable contents:
dst (string) — decoded destination group address (e.g. '1/1/4')
src (string) — decoded source individual address (e.g. '1.1.2')
type (string) — type of event, either "groupwrite", "groupresponse" or "groupread"
datahex (string) — data as a hex-encoded string which can be converted using data type functions
event.getvalue function can be used to get the event value if the event is bound to a known object.
Note: event variable is available only in Event-based functions, not in Resident and Scheduled.
Resident scripts
Resident scripts are executed infinite amount of times. Scripts are put into inactive state after each call and are resumed after delay timer expires.
Note: even though resident scripts are executed in parallel they should not have infinite loops or it will not be possible to reload scripts after editing.
Scheduled scripts
Scheduled scripts are executed when the system time matches the specified script start time. Scheduled script is run only once after each timer call.

2015 Schneider Electric. All rights reserved.

4 Modbus RTU and Modbus TCP

4.1 Characteristics

The Modbus open standard allows you to receive a more in-depth analysis of consumption in all areas of your building.
You can connect up to 31 Modbus slave devices of the following types of meters based on Modbus remote terminal unit (RTU) within one Modbus line:

- Schneider Electric energy meters
- Schneider Electric power meters
- Schneider Electric Smart Interface Modules (SIM10M module)
- Schneider Electric Smartlink
- Non-Schneider Electric Modbus TCP/RTU devices (offering you greater flexibility)

With the information which the homeLYnk provides, you can visualize energy or media consumption. This can also be used to reduce consumption through the use of control strategies within the KNX/IP network.

Modbus RTU is supported over RS485 interface. Modbus TCP is supported over Ethernet port. Modbus communication settings is done using *Modbus* table in homeLYnk *Configurator*. Modbus registers can be easily mapped using predefined Modbus profiles.

Modbus Master can be controlled directly from scripts (usually resident script is used to read Modbus values after some specific time interval and write them into KNX object or visualization). Once script is added, you can add the code in the Script Editor. There are lots of predefined code blocks in the *Helpers*.

NOTE: Do not use Modbus settings using profiles together with Modbus controlled from scripts. Interference of those two settings can cause communication errors. We strongly recommend you to use rather Modbus device profiles than configuration by scripting.

Application Example:

Requirements

- Measure and visualize how much energy is used for lighting an office building.
- Measure the gas and water consumption of the building.
- Monitor the quality of the network to ensure the operational safety of the IT equipment.

Solution

- Install an iEM3150 meter to measure the energy consumed by the lights.
- Install an iEM3255 meter to determine the power mains quality.
- Install a SIM10M module to measure gas and water consumption using pulse meters.
- Connect the devices to each other via Modbus.

2015 Schneider Electric. All rights reserved.

4.3 Modbus Device Profiles

There are plenty of ready to use device profiles, which are used for mapping the Modbus addresses (registers) to KNX group objects in homeLYnk. If there is a need to read/write some Modbus register, you only set the mapping rules, which allows you to access Modbus register by read/write of KNX group objects.

List of ready-to-use Modbus profiles in homeLYnk:

Modbus device	RTU scan	Modbus device	RTU scan
Compact_NSX-Compact_NSX_E	No	iEM-iEM3250	Yes
Masterpact_NT_NW-Masterpact_A	No	iEM-iEM3255	Yes
Masterpact_NT_NW-Masterpact_H	No	iEM-iEM3350	Yes
Masterpact_NT_NW-Masterpact_P	No	iEM-iEM3355	Yes
PM-PM1200	No	PM-PM710	No
PM-PM210	No	PM-PM750	No
PM-PM3250	Yes	PM-PM810	No
PM-PM3255	Yes	PM-PM820	No
PM-PM5110	No	PM-PM850	No
PM-PM5111	No	PM-PM870	No
PM-PM5310	No	PM-PM9C	No
PM-PM5330	No	SIM10M	No
PM-PM5350	No	Smartlink-RTU	Yes
iEM-iEM3150	Yes	Smartlink-TCP	No
iEM-iEM3155	Yes		

You can find Modbus devices connected to homeLYnk over Modbus RTU using scan function. This function is placed here: ***Configurator -> Modbus -> RTU scan.***

NOTE: Devices, which are marked as **"RTU scan = No"**, cannot be discovered using automatic scan function.

2015 Schneider Electric. All rights reserved.

4.3.1 New profile definition

If your Modbus device profile is not present in the list of ready-to-use profiles in homeLYnk *Configurator*, you can define your own profile.

Modbus device profiles are distributed in *.json files. You can use common text editor (e.g. Notepad) in order to create and edit your profile. While saving the file set extention to *.json. The following example shows the structure of new device profile:

{
"manufacturer": "Schneider Electric",
"description": "Example device",
"mapping": [
{ "name": "Output 1", "bus_datatype": "bool", "type": "coil", "address": 0, "writable": 1 },
{ "name": "Input 1", "bus_datatype": "float16", "type": "inputregister", "address": 0,
"value_multiplier": 0.001, "units": "V" }
]
}

Each line of "mapping" table of the json file contains mapping information of one Modbus register, coil, input or output. All the possible mapping settings are listed in the table below.

Parameter	**Description**	**Type**	**Required**
Name	Object name, e.g. Output 2	String	**Yes**
Bus_datatype	KNX object data type, key from dt table, e.g. float32	String / Number	**Yes**
Type	Modbus register type, possible values: coil, discreteinput, register, inputregister.	Strinf	**Yes**
Address	Register address (0-based)	Number	**Yes**
Writable	Set to true to enable writing to register if type is either coil or discreteinput	Boolean	No
Datatype	Modbus value data type. If set, conversion will be done automatically. Possible values: boolean, uint16, int16, float16, uint32, int32, float32, uint64, int64, quad10k, s10k	String	No
Value_delta	New value is sent when the difference between previously sent value and current value is larger than delta. Defaults to 0 (send after each read).	Number	No
Value_multiplier	Multiply resulting value by the specified number, value = value_base + value * value_multiplier.	Number	No
Value_bitmask	Bit mask to apply, shifting is done automatically based on least significant 1 found in the mask.	Number	No

2015 Schneider Electric. All rights reserved.

Value_nan	Array of 16-bit integers. If specified and read operation returns the same array no further processing of value is done.	Array	No
Value_conv	Apply one of built-in conversion functions	String (Internal)	No
Value_custom	Name of a built-in enumeration or a list of key -> value mapping, resulting value will be 0 if key is not found.	Strinfg / Object	No
Internal	Not visible to user when set to true, should be used for scale registers.	Boolean	No
Units	KNX object units/suffix	String	No
Address_scale	Address of register containing value scale, value = value * 10 ^ scale	Number	No
Read_count	Number of register to read at once (for devices that only support reading of a specific block of registers)	Number	No
Read_swap	Swap register order during conversion (endianness)	Boolean	No
Read_offset	Position of first register of data from the block of registers (0-based).	Number	No

Once you create your .json file, which contains all the information of your profile, you can upload it easily into your homeLYnk by ***Configurator -> Modbus -> Profiles ->Add profile.***

2015 Schneider Electric. All rights reserved.

4.4 Modbus RTU Interface

4.4.1 Characteristics of Modbus RTU Interface

- Supported over RS 485 physical interface
- Can act as Modbus/RTU Master or Modbus/RTU Slave
- Supported Function Codes: #01, #02, #03, #04, #05, #06, #07, #0F, #10
- Maximum 32 devices on bus
- RS 485 interface is not isolated!

4.4.2 RS 485 Bus Topology Characteristics

Main RS 485 Characteristics

- Mode of operation: differential
- Voltage at any bus terminal: -7 V to +12 V
- Receiver Input Sensitivity: +/-200 mV
- Sink/Source current: 60 mA
- Built-in asymmetrical protection against transient voltages resulting from electrostatic discharge (ESD), electrical fast transients (EFT), and lighting.
- Not isolated RS-485 interface

Connection Type	• point-to-point connections • point-to-multipoint connections
Type of Trunk Cable	shielded cable with 1 twisted pair and at least a third conductor
Maximum Length of Bus	1,000 m (3,280 ft) at 19,200 bit/s with the Telemecanique TSX CSA• cable
Maximum Number of Devices (without repeater)	32 (1 UL) devices, i.e. 31 slaves
Maximum Length of Tap Links	• 20 m (65 ft) for one tap link • a total of 40 m (131 ft) for all tap links available on the bus

Common Ground Wires

In order to keep the voltage between drivers and receivers within the allowed range (-7 V to +12 V), an additional third wire (in 2-wire systems) is required.
This wire will be used as common circuit and must therefore be directly connected to protective ground, preferably at one point only for the entire bus.
As grounding point for the entire bus you should choose the master device or its tap.

CAUTION: No terminal for cable shield. For longer cable in harsh environment we recommend you to place additional shield clamp close to the controller in order to drain EMC disturbances.

CAUTION: Earthed connectors from USB, RS232, LAN and Modbus are interconnected. Earth leakage currents may harm the operation of the controller.

Maximum Number of Devices without Repeater

An RS 485 network can principally have a maximum load of 32 devices.

2015 Schneider Electric. All rights reserved.

If you want to connect more than 32 devices to a standard RS 485 driver, then integrate a repeater in your network.

Biasing the Network

When there is no data activity on the Modbus bus, i.e. all nodes are in receive mode and there is no active driver available, the state of the line is unknown. In these cases the line is subjected to external noise or interference. In order to prevent the receivers from adopting improper states, the line needs to be biased, i.e. the constant state of the line must be maintained by an external pair of resistors connected to the RS 485 balanced pair.

RC Termination

To prevent unintended effects, like reflections, from occurring in your Modbus SL application, make sure to terminate the transmission lines properly.
Use RC termination to minimize the loop current and the line reflections. Furthermore RC termination increases the noise margin.
Choose two serial capacitors of 1 nF(10 V minimum) and two resistors of 120 Ω (0.25 W) as line termination. Integrate these components at both ends of your Modbus SL communication line.

R Termination Only

If the client insist on the R=150 Ohm termination only (not RC), he must connect external polarization resistors himself 450 - 650 Ohm (at the master's tap).
See the scheme in picture below.

Modbus interface isolation

Modbus interface of homeLYnk is not isolated. We strongly recommend to use optocoupler separation of Modbus line and homeLYnk RS485 interface. It will increase robustness of the Modbus network and reduce noise in the bus traffic.

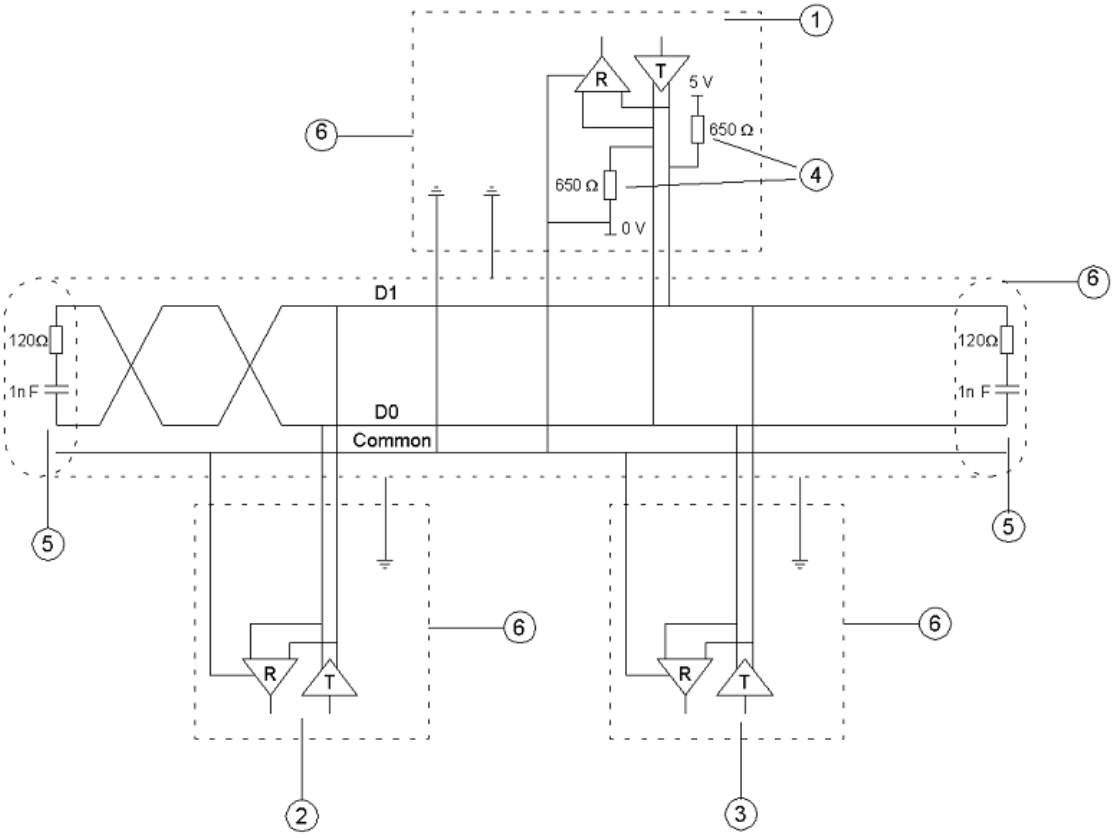
Termination, polarization and separation using Schneider Electric devices

Schneider Electric delivers devices for RS 485 termination, polarization and separation
Refer to following product numbers:

- TWD XCA ISO
- TWD XCA T3J

TWD XCA ISO and TWD XCA T3J devices can be used in order to ensure recommended RS 485 connection scheme (see the picture below). For more detailed information about TWD XCA ISO and TWD XCA T3J please refer to product documentation on Schneider Electric website.

2015 Schneider Electric. All rights reserved.



Elements of the application

No.	Element
1	master
2	slave 1
3	slave n
4	polarization resistors
5	line termination
6	shield

Maximum cable length without termination

When cable is not terminated, you must adjust the transmission rate to cable length.

Transmission Rate	9600	19,200	57,600	115,200
Bit Time (µs)	104.17	52.08	17.36	8.68
Bit Time/4 (µs)	26.0	13.0	4.3	2.2
Max. Cable Length Without Termination (m)	859	430	143	72

2015 Schneider Electric. All rights reserved.

4.4.3 Modbus RTU Configuration Commands

Create Modbus RTU object

```
require('luamodbus')
mb = luamodbus.rtu()
```

Open Modbus RTU connection

```
-- 19200 baud rate, even parity, 8 data bits, 1 stop bit, half duplex
mb:open('/dev/RS485', 19200, 'E', 8, 1, 'H')
mb:connect()
```

Terminal name:

'/dev/RS485'

Supported Baud rates:

300 bit/s
600 bit/s
1200 bit/s
2400 bit/s
4800 bit/s
9600 bit/s
19200 bit/s
38400 bit/s
57600 bit/s
115200 bit/s
230400 bit/s

Parity:

„N“ None
„E“ Even
„O“ Odd

Data bits and stop bits

Data bits: [Number of data bits = 5, 6, 7, **8**]
Stop bits: [Number of stop bits **1**, 2]

2015 Schneider Electric. All rights reserved.

Duplex:

„H“ Half duplex
“F” Full duplex (not supported in RS-485)

The Baud rate is set depending on the distance between Modbus RTU devices. For instance, with a Baud rate of 9600 bit/sec the maximum communication distance between 1 - 15 Modbus RTU device is 1,200 metres. With the Baud rate of 19200 bit/sec the maximum communication distance is 900 metres, as shown in the table:

Baudrate setting	Maximum communication distance for 1 to 15 Modbus RTU devices (Typical with Belden 3105A cables)
9600 bit/sec	1200 m
19200 bit/sec	900 m

Parity refers to the technique of checking if transmission has been successful when transmitting between the devices. It lets you know if some data has been lost during transmission.

Setting of Parity

The Modbus supports only 11 bit frames. "Parity" refers to the number of 1s in a given binary number. Odd parity means there are an odd number of 1s and even parity means that there is an even number of 1s. Parity bits are used as a means of error detection as digital data is transmitted and received.
Both the Gateway and Meter must always be set to the same as one another, odd, even or none. The default parity mode of Modbus is "even" parity.

- Parity = None: choose between one and two stop bits
- Parity = Even: one stop bit is set
- Parity = Odd: one stop bit is set

Delay Between Frames

Some devices require considerable time after the end of response until they are ready to receive the following request from the master. Henceforth, it applies to Schneider Electric SEPAM power devices and legacy slave devices. As they are slow in dealing with the original request they may miss the following request.
The time between the requests should be greater than 3.5 characters according to the Modbus specification. However, these legacy devices need more time. Please use delay command appropriately:

```
--Wait for 1.5 seconds
os.sleep(1.5)
```

Communication itself takes care of minimal 3, 5 character delay.

Set slave address:

```
--set slave address to 123
mb:setslave(123)
```

[1..247]

2015 Schneider Electric. All rights reserved.

Read registers:

--read from address 1000 and write it to value
value = mb:readregisters(1000)

Close modbus connection:

mb:close()

Example:

```
--init modbus on first script execution
    if not mb then
        require('luamodbus')
        mb = luamodbus.rtu()
        mb:open('/dev/RS485', 38400, 'E', 8, 1, 'H')
        mb:connect()
    end
    mb:setslave(30)
    mb:flush()
```

Timeout interval between two consecutive bytes of the same message

```
    mb:getbytetimeout()
    mb:setbytetimeout(timeout)
```

Timeout interval used to wait for a response:

```
    mb:getresponsetimeout()
    mb:setresponsetimeout(timeout)
```

Timeout interval used to for an incoming indication from master (slave mode only):

```
    mb:getreceivetimeout()
    mb:setreceivetimeout(timeout)
```


2015 Schneider Electric. All rights reserved.

4.5 Modbus TCP Interface

4.5.1 Characteristics of Modbus TCP Interface

- Supported over Ethernet interface 10Mb, 100Mb
- Can act as Modbus/TCP-IP Client or Modbus/TCP-IP Server
- DHCP support
- Max. opened TCP connections: 100
- Supported Function Codes: #01, #02, #03, #04, #05, #06, #07, #0F, #10

Grounding-Isolation:

- Metal cover of the RJ45 socket is connected to device ground

4.5.2 Modbus TCP configuration commands

Create Modbus TCP object

```
require('luamodbus')
mb = luamodbus.tcp()
```

Open Modbus TCP connection

```
-- IP: 192.168.1.2, port: 1234
mb:open('192.168.1.2', 1234)
mb:connect()
```

All the rest of commands needed to configure the Modbus TCP connection are the same as for Modbus RTU.

2015 Schneider Electric. All rights reserved.

4.6 Modbus Function Codes and Corresponding Master Functions

All the possible Modbus function codes, which can be used in homeLYnk, are listed below. There is a Lua function in homeLYnk for each function code.

All the functions described bellow can be used both for Modbus TCP and Modbus RTU.

FC#01 Read Coils:

Name	**"Read single coil"**
Command	**coil = mb:readcoils(address)**
Arguments	[address]: address of the coils
Returned values	1: ON, 0: OFF
Exception codes:	01 or 02 or 03 or 04

Name	**"Read Multiple coil"**
Command	**coil= mb:readcoils(start, count)**
Arguments	[start]: address of first coil to read [count]: number of coils to read (max 2000)
Returned values	1: ON, 0: OFF
Exception codes	01 or 02 or 03 or 04
Example	*coil1,coil2,coil3= mb:readcoils(1000, 3)* Value read from coil address 1000 is returned into variable coil1. Value read from coil address 1001 is returned into variable coil2. Value read from coil address 1002 is returned into variable coil3.

FC#02 Read Discrete Inputs:

Name	**"Read discrete input"**
Command	**value = mb:readdiscreteinputs(address)**
Arguments	[address]: address of the input
Returned values	1: ON, 0: OFF
Exception codes:	01 or 02 or 03 or 04

Name	**"Read discrete inputs"**
Command	**value = mb:readdiscreteinputs(start,count)**
Arguments	[address]: address of first input to read [count]: number of inputs to read (max 2000)
Returned values	1: ON, 0: OFF
Exception codes:	01 or 02 or 03 or 04
Example	*bool1, bool2= mb: readdiscreteinputs(10,2)* Value read from discrete input address 11 is returned into variable bool1. Value read from discrete input address 12 is returned into variable bool2.

2015 Schneider Electric. All rights reserved.

FC#03 Read Holding Registers:

Name	**"Read registers"**
Command	value = mb:readregisters(address,count)
Arguments	[address]: address of first register to read [count]: number of registers to read (max 125)
Returned values	2byte values
Exception codes:	01 or 02 or 03 or 04
Example	*int1, int2= mb: readregisters(1100,3)* Value read from register address 1100 is returned into variable int1. Value read from register address 1101 is returned into variable int2. Value read from register address 1102 is returned into variable int3.

FC#04 Read Input Registers:

Name	**"Read input registers"**
Command	value = mb:readinputregisters(address,count)
Arguments	[address]: address of first input register to read [count]: number of input registers to read (max 125)
Returned values	2byte values
Exception codes:	01 or 02 or 03 or 04
Example	*value1, value2, value3, value4 = mb:readinputregisters(1015,4)* Value read from input register address 1015 is returned into variable value1. Value read from input register address 1016 is returned into variable value2. Value read from input register address 1017 is returned into variable value3. Value read from input register address 1018 is returned into variable value4.

FC#05 Write Single Coil:

Name **"Write single bit"**
Command **value = mb:writebits(1000, value)**
[starting address, value "true" or "false"/"0"]

FC#06 Write Single Register:

Name **"Write single register"**
Command **value = mb:writeregisters(1000, 123)**
[address, value]

FC#0F Write Multiple Coils:

Name **"Write multiple bits"**
Command **value = mb:writebits(1000, true, false,true,…)**
[address, bit value1, bit value2,..{max 1968 bits}]

2015 Schneider Electric. All rights reserved.

FC#10 Write Multiple Registers:

Name **“Write multiple registers”**
Command **value = mb:writeregisters(1000, 123, 321,222,..)**
[address, value1, value2, ..{max 123 registers}]

Exception codes

mb:readcoils(start, count)

mb:readdiscreteinputs(start, count)

mb:readregisters(start, count)

mb:readinputregisters(start, count)

These commands read one or more registers/coils from the start address and return all values in case of success. In case of error, three variables are sent back:

- Nil
- Exception code description
- Exception code

2015 Schneider Electric. All rights reserved.

The following information is taken from the Modicon Web site (http://modbus.org) and the Modbus application protocol manual.

MODBUS Exception Codes		
Code	Name	Meaning
01	Illegal Function	The Function Code received in the query is not an allowable action for the server (or slave). This may be because the function code is only applicable to newer devices, and was not implemented in the unit selected. It could also indicate that the server (or slave) is in the wrong state to process a request of this type, for example because it is not configured and is being asked to return register values.
02	Illegal Data Address	The data address received in the query is not an allowable address for the server (or slave). More specifically, the combination of reference number and transfer length is invalid. For a controller with 100 registers a request of offset 96 and a length of 5 will generate exception 02.
03	Illegal Data Value	The value contained in the query data field is not an allowable value for the server (or slave). This indicates a fault in the structure of the remainder of a complex request, such as that the implied length is incorrect. It specifically does NOT mean that a data item submitted for storage in a register has a value outside the expectation of the application program, since the MODBUS protocol is unaware of the significance of any particular value of any particular register.
04	Failure In Associated Device	An Unrecoverable error occurred while the server (or slave) was attempting to perform the requested action. (See Note 1)
05	Acknowledge	Specialized in conjunction with programming commands. The server (or slave) has accepted the request and is processing it, but long duration of time will be required to do so. This response is returned to prevent a timeout error from occurring in the client (or master). The client (or master) can next issue a poll program complete message to determine if processing is completed.
06	Busy, Rejected Message	Specialized use in conjunction with programming commands. The server (of slave) is engaged in processing a long-duration program command. The client (or master) should retransmit the message later when the server (or slave) is free.
07	NAK – Negative Acknowledgement	The program function just requested cannot be performed. Issue poll to obtain detailed device dependent error information. Valid for Program/Poll 13 and 14 only.
08	Memory Parity Error	Specialized use in conjunction with function codes 20 and 21 and reference type 6, to indicate that the extended file area failed to pass a consistency check. The server (or slave) attempted to read record file, but detected a parity error in the memory. The client (or master) can retry the request, but service may be required on the server (or slave) device.

2015 Schneider Electric. All rights reserved.

0A	Gateway Path Unavailable	Specialized use in conjunction with gateways. Indicates that the gateway was unable to allocate an internal communication path from the input port to the out port for processing the request.
0B	Gateway Target Device Failed to respond	Specialized use in conjunction with gateways. Indicates that no response was obtained from the target device. Usually means that the device is not present on the network.

4.7 Modbus Master Functions

Functions listed below can be used for Modbus RTU master or Modbus TCP client.

mb:setslave(slaveid)
sets slave id to read/write data from/to

mb:readcoils(start, count) [01]
start – address of first coil to read
count – number of coils to read

mb:readdiscreteinputs(start, count) [02]
start – address of first discrete input to read
count – number of discrete inputs to read

mb:readregisters(start, count) [03]
start – address of first holding register to read
count – number of holding registers to read

mb:readinputregisters(start, count) [04]
start – address of input register to read
count – number of input registers to read
returns all values on success and nil, error description on error

mb:writebits(start, v1, [v2, [v3, ...]]) [05]
writes values to coils from start address

mb:writeregisters(start, v1, [v2, [v3, ...]]) [06]
writes values to registers/coils from the start address
single write will be used when only one value is supplied, multiple write otherwise
returns all of values written on success and nil, error description on error

mb:reportslaveid()
reads slave internal data
returns values on success
returns nil, error description on error

2015 Schneider Electric. All rights reserved.

4.8 Modbus slave functions

4.8.1 Receive data from master

mb:receive()
receives data from master with 1 minute timeout
returns data as a binary string on success
returns nil, error description on error

4.8.2 Set modbus mapping of slave device

mb:setmapping(coils, inputs, holding_regs, input_regs)
creates memory mapping for the registers with size specified for each type

4.8.3 Handle slave

mb:handleslave()
waits for an incoming indication from master and sends a reply when necessary

4.8.4 Get functions

mb:getcoils(start, count)

mb:getdiscreteinputs(start, count)

mb:getinputregisters(start, count)

mb:getregisters(start, count)
gets one or many register/coil/input values from mapping from the start address
returns all values on success
returns nil, error description on error, exception code if applicable

4.8.5 Set functions

mb:setcoils(start, v1, [v2, [v3, ...]])

mb:setdiscreteinputs(start, v1, [v2, [v3, ...]])

mb:setinputregisters(start, v1, [v2, [v3, ...]])

mb:setregisters(start, v1, [v2, [v3, ...]])
sets values to register/coil mapping from the start address
returns true on success
returns nil, error description on error, exception code if applicable

4.8.6 Callback functions

mb:setwritecoilcb(fn)

mb:setwriteregistercb(fn)
sets a callback function for coil/register write event

2015 Schneider Electric. All rights reserved.

callback should accept two parameters - coil/register address and value (boolean or number)
for multiple writes callback is executed for each coil/register separately
use nil to remove a callback.

Note: For more details about Modbus slave settings refer to application note document ***AN_016_homeLYnk_as_a_Modbus_slave***.

2015 Schneider Electric. All rights reserved.

5 Port Forwarding

5.1 Introduction

Port forwarding is used to get remote access to IP device on local network, like homeLYnk. Settings have to be done in the network router. Manual of the particular router explains, how to set port forwarding. In case of issues, contact of the technical support of the router provider may be needed.

5.2 HTTP and HTTPS

homeLYnk uses two possible ways of connection.

5.2.1 HTTP

Default one is through HTTP and port 80. HTTP is not encrypted and is not a secured way of connection. This connection is safe to use on local network, but not recommended to use for remote connection. If this is selected, then in the router, port 80 has to be forwarded with the IP of the homeLYnk.

To connect to homeLYnk using port forwarding with HTTP connection, following has to be entered in the web browser address bar:
HTTP://IP:Port

Where IP is an IP of the internet connection of the house. This information can be found inside the router or the contact the internet provider support.

5.2.2 HTTPS

HTTPS is a secured and an encrypted connection, and is strongly recommended to be used as a remote connection. Using the secure connection, port 443 has to be forwarded in the router.

To remotely connect through the secured HTTPS connection, following has to be entered in the web browser address bar:
HTTPS://IP:Port

Where IP is an IP of the internet connection of the house. This information can be found inside the router or the contact the internet provider support.

NOTE: Apple devices with OS7.0 and above using the remote connection must forward (port+1) for correct status feedback in visualization. For port 80 it would be feedback port 81.
If using a custom port A, you need to forward port A to homeLYnk's port 80, and port A + 1 to homeLYnk's port 81. For example, if user wants remote access to visualization and uses port 1234 to access his HL, he must forward port 1234 to homeLYnk's port 80, port 1235 to homeLYnk's port 81.

NOTE: If you want to use different port number than default ports 80 and 443, you can set the additional ports in *Configurator -> Utilities -> System -> Services -> HTTP server*.

2015 Schneider Electric. All rights reserved.

6 BACnet

6.1 Characteristics

BACnet is a communication protocol for *Building Automation and Control Networks*. It is an ASHRAE – *American Society of Heating, Refrigerating and Air-Conditioning Engineers*, ANSI – *American National Standards Institute*, and ISO – *International Organization for Standardization* protocol.

homeLYnk has been certified by BACnet Testing Laboratories (BTL) as BACnet Application Specific Controller (B – ASC).

BACnet is designed to allow communication of building automation and control systems for application such as heating, ventilation, air conditioning control, lighting control, access control, fire detection systems and their associated equipment. BACnet protocol provides exchange information for building automation devices, regardless of the particular building service they perform.

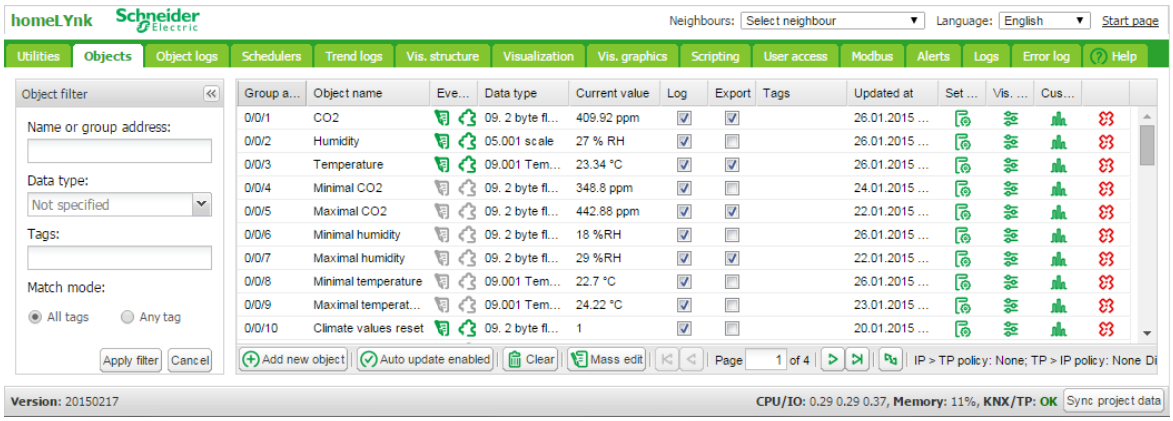
6.2 homeLYnk Configuration

Interconnection of homeLYnk and other BACnet device is done over Ethernet physical layer. homeLYnk can act as a BACnet server only. It means that homeLYnk serves data which can be read by BACnet client device and BACnet client device can write data to the server.

As homeLYnk is KNX based device the connection to BACnet network comes from KNX group objects, which are exported to BACnet.

6.3 Object export

All the KNX objects in homeLYnk object list (***Configurator → Objects***) has the parameter “Export”. By selecting this “Export” checkbox the specific KNX object will be visible in BACnet as BACnet object.



Binary objects will appear as binary values, numeric values will appear as analogue values. Other data types are not supported.

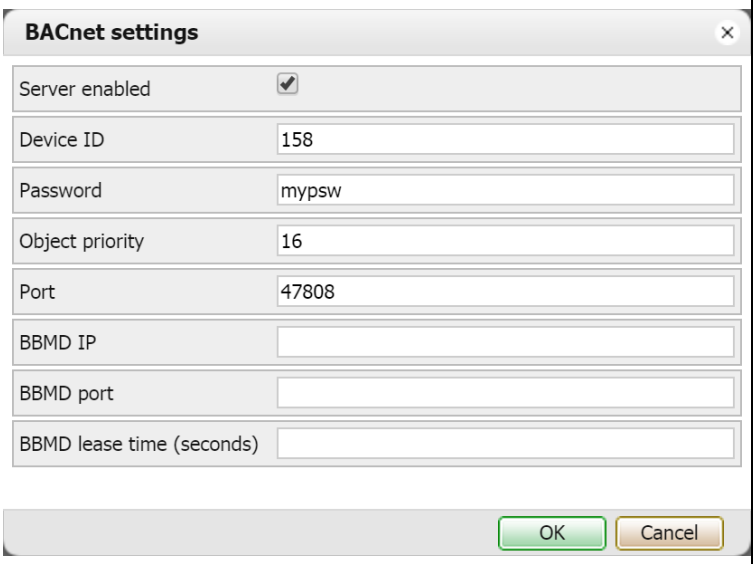
2015 Schneider Electric. All rights reserved.

6.4 BACnet Configuration

BACnet configuration consists of setting BACnet server parameters in homeLYnk. The BMS - *Building Management System* discovers the exposed data.

6.4.1 Configuration

homeLYnk acts as a BACnet server which has to be configured under ***Configurator → Utilities → System → Network → BACnet settings***



Server enabled – Enable/Disable BACnet server

Device ID – BACnet device ID which must be unique on BACnet network

Password – BACnet password

Objects priority – Define to which priority array homeLYnk will write to. By default, homeLYnk does not write to priority array only to Relinquish Default Property

Port – BACnet port, default 47808

BBMD IP – BACnet router IP.

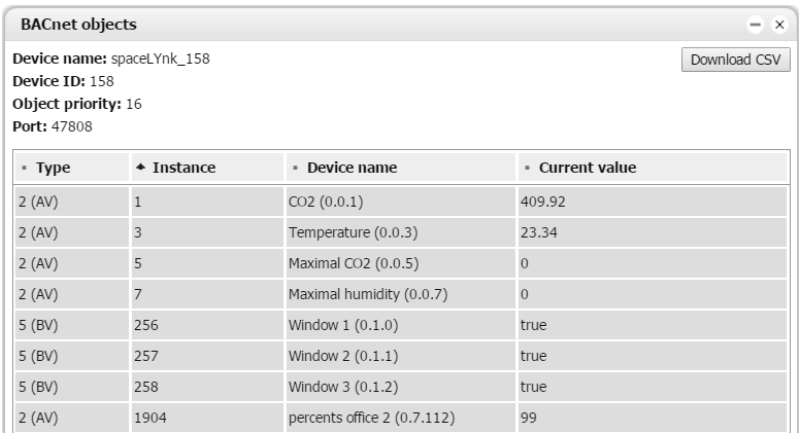
BBMD port – BACnet router port

BBMD lease time (seconds) – registration resend interval

6.4.2 BACnet objects

List of BACnet objects with its parameters is available under:

Configurator → Utilities → System → Network → BACnet objects



Type	Instance	Device name	Current value
2 (AV)	1	CO2 (0.0.1)	409.92
2 (AV)	3	Temperature (0.0.3)	23.34
2 (AV)	5	Maximal CO2 (0.0.5)	0
2 (AV)	7	Maximal humidity (0.0.7)	0
5 (BV)	256	Window 1 (0.1.0)	true
5 (BV)	257	Window 2 (0.1.1)	true
5 (BV)	258	Window 3 (0.1.2)	true
2 (AV)	1904	percents office 2 (0.7.112)	99

Device name is combined from Hostname and Device ID

BACnet objects can be downloaded to CSV file via Download CSV button

NOTE: Download CSV button is hidden, if browser does not support this feature.

NOTE: Maximum number of BACnet objects is 150.

2015 Schneider Electric. All rights reserved.

6.5 BACnet Standardized Device Profile

homeLYnk has been tested at the BACnet Testing Labs (BTL) and found to comply with all the necessary interoperability requirements.

More details and results from BTL testing can be found here:

http://www.bacnetinternational.net/catalog/index.php?m=20&p=1201

6.5.1 List all BACnet Interoperability Building Blocks (BIBBs) Supported

Data Sharing	ReadProperty-B	DS-RP-B
	ReadPropertyMultiple-B	DS-RPM-B
	WriteProperty-B	DS-WP-B
	COV-B	DS-COV-B
Device and Network Management	Dynamic Device Binding-B	DM-DDB-B
	Dynamic Object Binding-B	DM-DOB-B
	DeviceCommunicationsControl-B	DM-DCC-B
	TimeSynchronization-B	DM-TS-B
	UTCTimeSynchronization-B	DM-UTC-B
	ReinitializeDevice-B	DM-RD-B

6.5.2 BACnet Object Types Supported

- Device
- Analog Value
- Binary Value

6.5.3 Data Link Layer Options

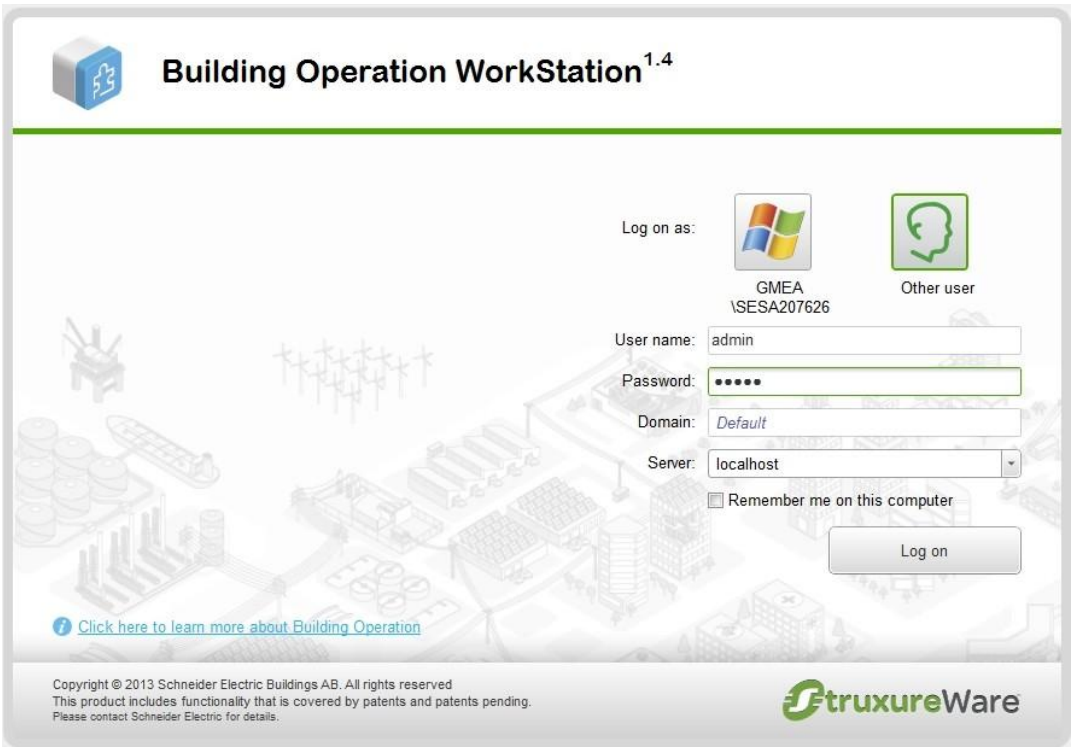
- Media: BACnet IP
    - Option: Register as a Foreign Device

2015 Schneider Electric. All rights reserved.

6.6 Building Operation Workstation

Schneider Electric StruxureWare is a BACnet certified Building Management System.
Building Operation WorkStation is a software used to configure and commission Enterprise Server and the Automation Server which can retrieve and send data to homeLYnk.

There is a dedicated document, which describes the interoperability between homeLYnk and Building Operation Workstation over BACnet. If you look for more details about this topic please refer to application note ***AN001_homeLYnk integration using BACnet_v1.4.pdf***.



2015 Schneider Electric. All rights reserved.

7 RS-232 Serial Line

7.1 Characteristics

The RS-232 serial interface communication standard has been in use for many years. It is one of the most widely used connections for serial data transmitting because it is simple and reliable.

The RS232 serial interface standard still retains its popularity and remains in widespread use. It is still found on some computers and on many interfaces, often being used for applications ranging from data acquisition to supply a serial data communications facility in general computer environments.

The long term and widespread use of the RS232 standard has meant that products are both cheap and freely available, and in these days of new higher speed standards, the reliable, robust RS232 standard still has much to offer. The interface is intended to operate over distances of up to 15 meter; it is based on one Master/ one Slave rule.

Application Example:

- Connection to simple devices or other bus sub systems.
- Audio/video, IR system integration.

7.2 Configuration Commands

Open connection:

```
require('serial')
port = serial.open('/dev/RS232', {baudrate = 9600})
```

Write to port:

```
port:write('test data')
```

Blocking read:

```
-- script will block until 10 characters are read
data = port:read(10)
```

Timeout read:

```
-- script will wait for 10 characters for 20 seconds

data = port:read(10, 20)
```

Close serial port:

```
port:close()
```


2015 Schneider Electric. All rights reserved.

NOTE: RS-485 serial line is controlled in the same way using the same Configuration Commands as mentioned above. The only diffetend is in the serial.open command:
port = serial.open('/dev/RS485', {baudrate = 9600})

NOTE: For more details about RS-232 communication please refer to application note ***AN010_RS232 control with homeLYnk_v1.0.pdf***

2015 Schneider Electric. All rights reserved.

8 USB 2.0

8.1 Characteristics

- USB 2.0 provides a bandwidth of 480 Mbit/s, corresponding to an effective image data rate of 40 MB/s.
- Integrated voltage supply (5 VDC) for devices in the 4-pole cable. Devices complying with the USB specification may consume a total of 500 mA from the bus. Devices with a power of up to 2.5 W can therefore be supplied via the bus.
- USB cable must only be 4.5 m long at the maximum.
- Data transmission is possible in both directions

Application Example:

USB interface can be used for extending memory capacity via attaching USB flash drive.

8.2 Configuration Commands

io.readfile (file)

Read whole file at once. Returns file contents as a string on success or nil on error.

io.writefile (file, data)

Writes given data to a file. Data can be either a value convertible to string or a table of such values. When data is a table, then each table item is terminated by a new line character. Return Boolean as write result when file can be open for writing or nil when file cannot be accessed.

NOTE: USB flash drive supports FAT, FAT32 and NTFS file system. Maximum size of Flash drive is 32GB.

8.3 Send and receive SMS messages via attaching USB GSM adapter.

- Use Huawei E173 modem.
- The modem has to be plugged into USB port of homeLYnk and it starts operating immediately.
- Specific functions should be added into user script library with PIN code setting and telephone number white-list which will be able to receive and send SMS messages.

2015 Schneider Electric. All rights reserved.

8.3.1 Command syntax:

In order to change object value using sms or read value of object by SMS request, you have to send sms in the format described below.

Write to bus:

- SMS command format: W ALIAS VALUE

Read from bus:

- SMS command format: R ALIAS
- On read request, script will reply with SMS message containing current value of selected object

ALIAS can be:

- Group address (e.g. 1/1/1)
- Name (e.g. Obj1). If name contains homes, then it must be escaped using double quotes (e.g. “Room Temperature”)

NOTE: Object data type and name must be set in ***Configurator -> Objects tab***. Otherwise, script will not be able to read and write to object.

NOTE: Only ASCII symbols are accepted in the message.

NOTE: For more details about sending SMS please refer to application note document ***AN011_Email SMS and FTP in homeLYnk_v1.0.pdf***.

2015 Schneider Electric. All rights reserved.

9 Block Programming

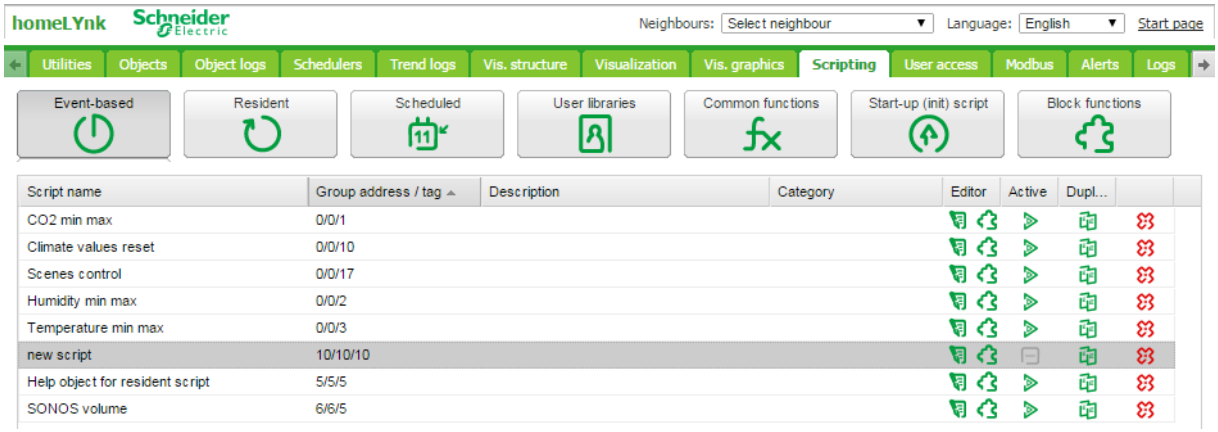
9.1 Introduction

homeLYnk programming can be done either using Lua scripts or block programming. For those who are programming beginners, it is recommended to start with block programming. It is based on Lua scripting but visualized in more friendly way.

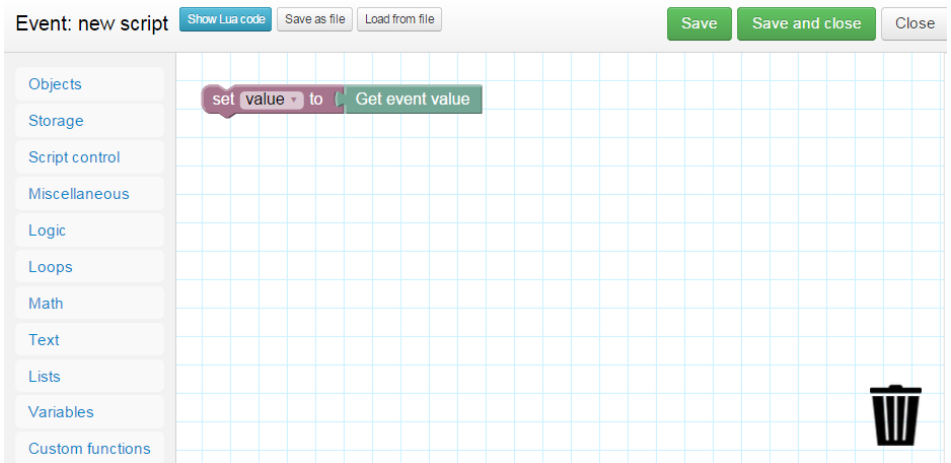
9.2 Block programming interface

In order to create blocks, enable this functionality in ***Utilities → General configuration → Enable Block Editor*.**

Once the script is added, you can see puzzle icon to access Block editor.



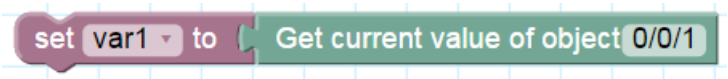
Blocks are sorted by categories on the left side. Each block is puzzle based and can be put only in appropriate location / other block.



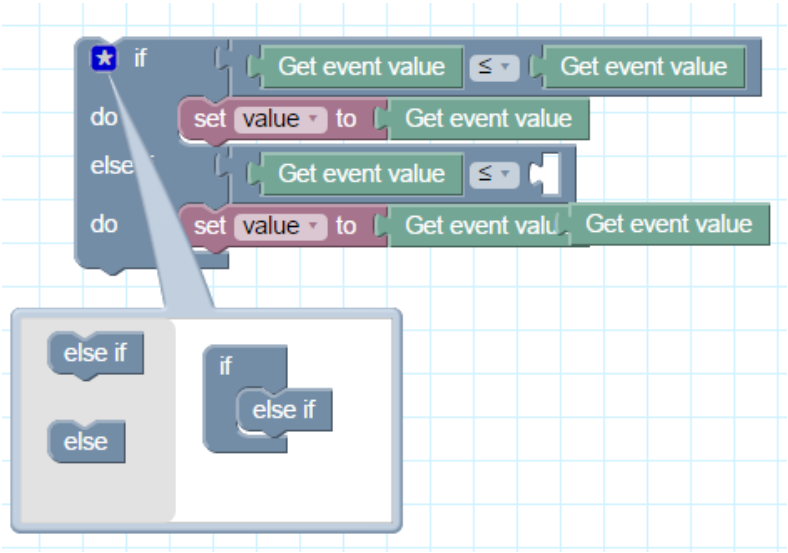
2015 Schneider Electric. All rights reserved.

Hole in the right side of the block denotes input of the block. Left side segment of the block denotes output of the block. See example below:

- Block “set var1 to” has input on the right side, where it reads value to be written into variable var1.
- Block “Get current value of object 0/0/1” has output on the left side, where the current value of object 0/0/1 is returned.
- When the blocks are given together, variable var1 is set to current value of object 0/0/1.



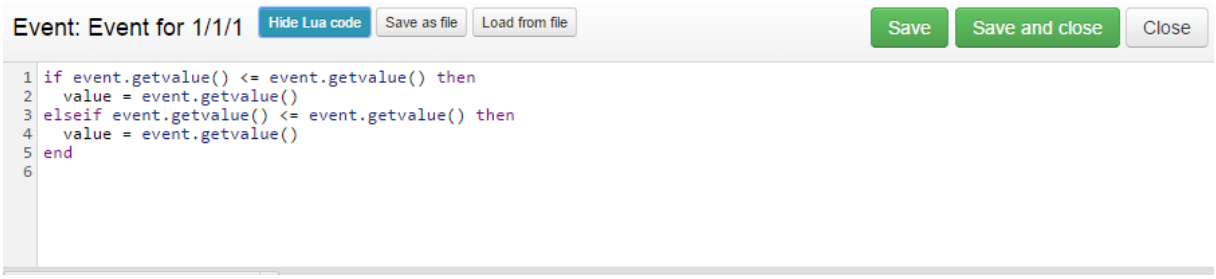
If the block is indicated with the blue label on the top left corner, you can define the structure of the block (e.g.”if..do..else”).



Click right-mouse button and select “*Delete Block*” or drag the block to the basket if you want to delete it

You can always look at the LUA code by clicking on *Show/Hide Lua code* button. This will allow you to learn the scripting language.

2015 Schneider Electric. All rights reserved.



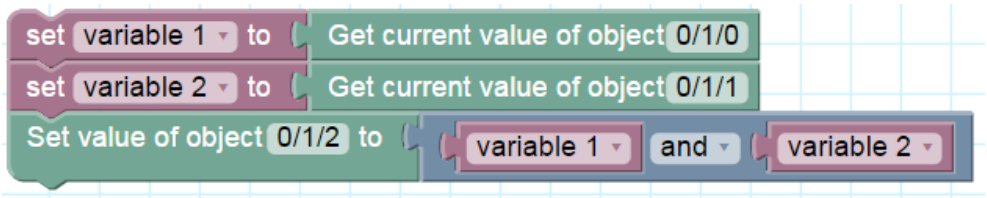
NOTE: Script created in function blocks can be transferred into Lua script, but it cannot be transferred back into block function script. In order to transfer block function script into Lua script, save and close your script and open it in script editor using this icon .

9.3 Usage Examples

This subchapter shows how to use function blocks correctly. Examples with description help to better understand the concept of block programming.

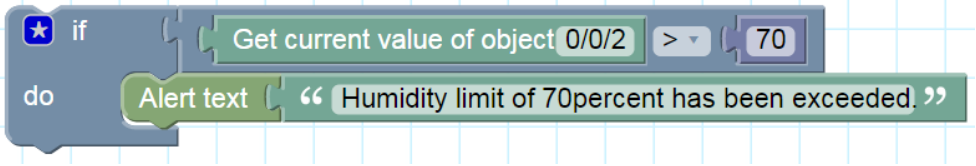
9.3.1 AND of two KNX objects

Current value of object 0/1/0 is read and saved into variable named “variable 1”.
Current value of object 0/1/1 is read and saved into variable named “variable 2”.
Logical function AND is performed with variables “variable 1” and “variable 2”.
The logical result is saved into KNX object 0/1/2.


set variable 1 to Get current value of object 0/1/0
set variable 2 to Get current value of object 0/1/1
Set value of object 0/1/2 to variable 1 and variable 2

9.3.2 IF object value exceed maximum, alert is created.

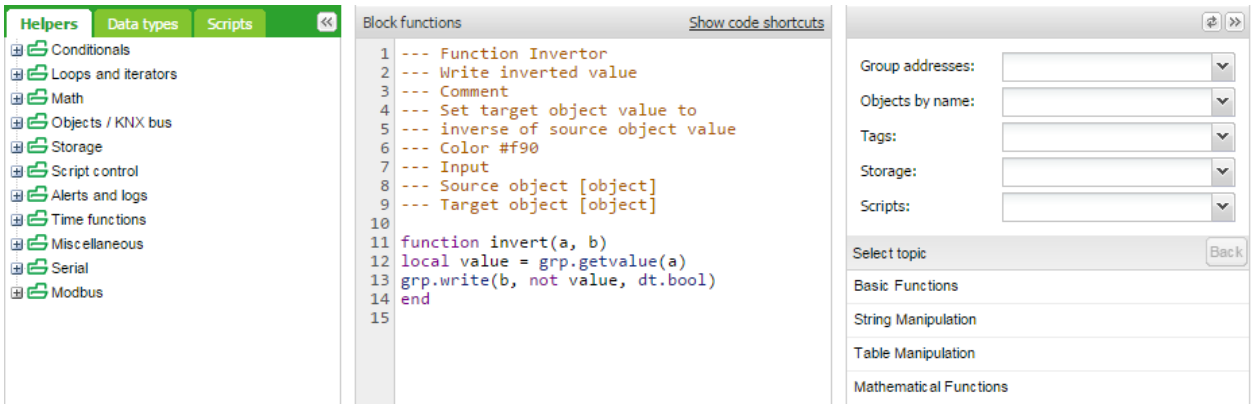
Current value of object 0/0/2 is read.
Current value of object 0/0/2 is compared with static value 70.
If value of object 0/0/2 is greater than 70, alert message is created.



2015 Schneider Electric. All rights reserved.

9.4 Custom Function Blocks

In Scripting menu there is ***Block functions button***. Here you can create custom block functions which can be later used as ready block in Block editor.



Each function must have a special comment. Remember, that special keywords (**Function, Comment, Input**, optionally **Color**) must be used in the function comment in order to create the block function. The keywords are highlighted in the description below.

Function block comment:

1. First line must have **Function** keyword followed by the function name
2. Second line contains short function description which is shown as block title
3. If third line contains **Comment** keyword, all following lines until **Input** will be added to block comment tooltip.
4. Optionally, block color may be specified in hexadecimal format (#f00 or #ff9900) or numeric format as hue value between 0 and 359.
5. Keyword **Input** specifies that all the following lines will list the description of inputs.
6. Following lines contain input list. Each block can have any number of inputs. Inputs are a function parameter
7. If input definition has **[object], [storage]** or **[tag]** in its name then the input is replaced with object, storage or tag selection input.

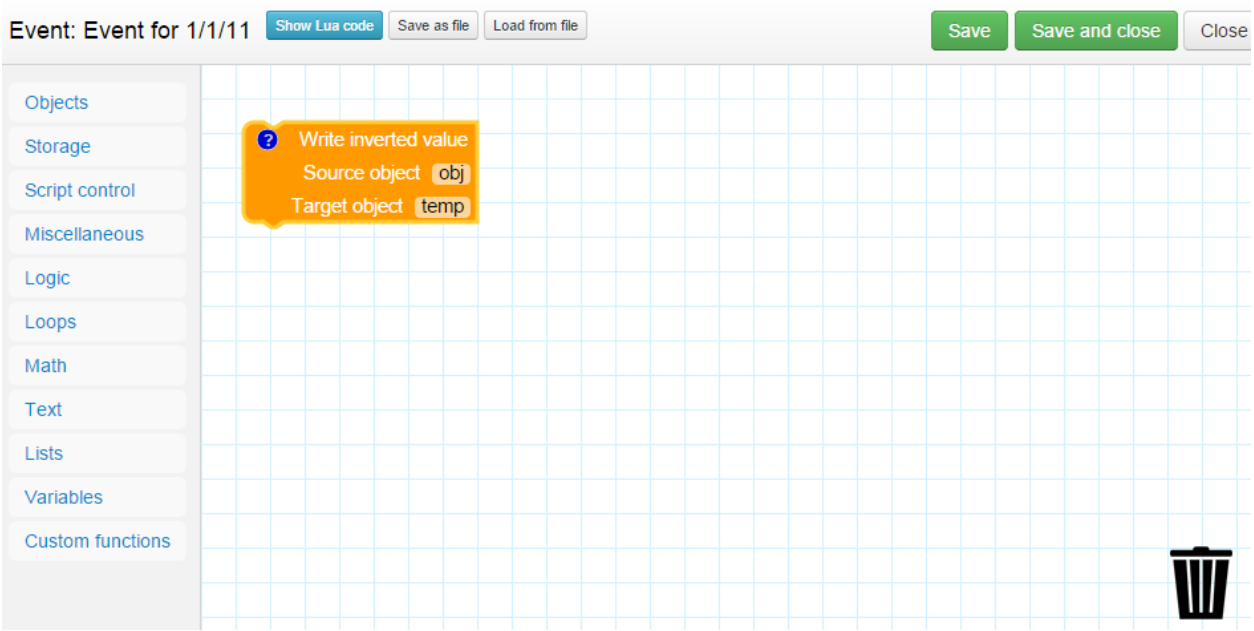
Each output variable is assigned to the corresponding function return value.

2015 Schneider Electric. All rights reserved.

Example:

*--- **Function** invert*
--- Write inverted value
*--- **Comment***
--- Set target object value to
--- inverse of source object value
*--- **Color** #f90*
*--- **Input***
*--- Source object **[object]***
*--- Target object **[object]***
function invert(a, b)
local value = grp.getvalue(a)
grp.write(b, not value, dt.bool)
end

Once block function is added, it is available as a block in Block editor in section Custom functions.



2015 Schneider Electric. All rights reserved.

9.5 Special Function Blocks

There is number of special function blocks, which are placed in the Custom functions section. These function blocks are very easy to use even without any programming experience.

More detailed description of function blocks listed below can be found in application note document ***AN024_Block functions for spaceLYnk and homeLYnk controller.pdf***.

9.5.1 General

- Process kill (PID)
- Scene
- Write with delay

9.5.2 Convertors

- Date (time) to string

9.5.3 Logical

- AND
- OR

9.5.4 Statistical

- AVG
- MIN
- MAX

2015 Schneider Electric. All rights reserved.

10 LUA – Programming Language

LUA is a powerful, fast, lightweight, embeddable scripting language. LUA combines simple procedural syntax with powerful data description constructs based on associative arrays and extensible semantics. LUA is dynamically typed, runs by interpreting byte code for a register-based virtual machine, and has automatic memory management with incremental garbage collection, making it ideal for configuration, scripting, and rapid prototyping.

Programming in LUA as scripting language for homeLYnk is primary based on the writing functions. Functions in LUA are ***first-class values*** with proper ***lexical scoping***.

What does it mean for functions to be ***first-class values***? It means that, in LUA, a function is a value with the same rights as conventional values like: numbers and strings. Functions can be stored in variables (both global and local) and in tables. They can be passed as arguments, and can be returned by other functions.

What does it mean for functions to have ***lexical scoping***? It means that the functions can access variables of its enclosing functions.

10.1 Object Functions

The ***grp*** provides simplified access to objects stored in the database and group address request helpers.

Most functions use ***alias*** parameter — object group address or unique object name. (e.g. '1/1/1' or 'My object')

grp.getvalue(alias)

Returns value for the given alias or LUA ***nil*** when object cannot be found.

grp.find(alias)

Returns single object for the given alias. Object value will be decoded automatically only if the data type has been specified in the 'Objects' module. Returns LUA ***nil*** when object cannot be found, otherwise, it returns LUA *table* with the following items:

- ***address*** — object group address
- ***updatetime*** — latest update time in UNIX timestamp format. Use LUA ***os.date()*** to convert to readable date formats

When object data type has been specified in the 'Objects' module the following fields are available:

- ***name*** — unique object name
- ***datatype*** — object data type as specified by user
- ***decoded*** — set to ***true*** when decoded value is available
- ***value*** — decoded object value

2015 Schneider Electric. All rights reserved.

grp.tag(tags, mode)

Returns LUA ***table*** containing objects with the given tag. Tags parameter can be either LUA *table* or a string. Mode parameter can be either '***all***' (return objects that have all of the given tags) or '***any***' (***default*** — returns objects that have any of the given tags). You can use ***Returned object functions*** on the returned table.

grp.alias (alias)

Converts group address to object name or name to address. Returns LUA ***nil*** when object cannot be found.

10.2 Group Communication Functions

These functions should only be used if it is required to access objects by group address directly, it is recommended to use single or multiple object functions.

grp.write (alias, value, datatype)

Sends group write requests to the given alias. Data type is taken from the database if not specified as third parameter. Returns LUA ***boolean*** as the result.

grp.response (alias, value, datatype)

Similar to ***grp.write*****.** Sends group response request to the given alias.

grp.read(alias)

Sends group read requests to the given alias.

NOTE: This function returns immediately and cannot be used to return the result of read request. Use event-based script instead.

grp.update(alias, value, datatype)

Similar to ***grp.write***, but does not send any value to KNX TP bus. It only sends telegrams to KNX IP, when KNX IP features are turned on. Useful for objects that are used in visualization only.

10.3 Returned object functions

Objects received by using **grp.find(alias)** ***or*** **grp.tag(tags, mode)** have the following functions attached to them:

Always check that the returned object was found otherwise calling these functions will result in an error. See the example below.

object:write(value, datatype)

Sends group write requests to object's group address. Data type is taken from the database if not specified as second parameter. Returns LUA ***boolean*** as the result.

object:response(value, datatype)

Similar to ***object:write***. Sends group response request to object's group address.

object:read()

Sends group read requests to object's group address.

2015 Schneider Electric. All rights reserved.

NOTE: This function returns immediately and cannot be used to return the result of read request. Use event-based script instead.

object:update(value, datatype)

Similar to ***object:write***, but *does not send* new value to the bus. Useful for objects that are used in visualization only.

10.4 Data type Functions

knxdatatype

Object provides data encoding and decoding between LUA and KNX data formats.

knxdatatype.decode(value, datatype)

Converts hex-encoded data to LUA variable based on given data type. Data type is specified either as KNX primary data type (integer between 1 and 16) or a secondary data type (integer between 1000 and 16000). Return values:

- success — decoded data as LUA variable (type depends on data type), value length in bytes
- error — nil, error string

10.5 Data Types

The following data types can be used for encoding and decoding of KNX data. Data representation on LUA level and predefined constants (in bold) is given below:

- ***bool 1 bit (boolean) - dt.***— boolean
- ***2 bit (1 bit controlled) - dt.bit2*** — number
- ***4 bit (3 bit controlled) - dt.bit4*** — number
- ***1 byte ASCII character - dt.char*** — string
- ***1 byte unsigned integer - dt.uint8*** — number
- ***1 byte signed integer - dt.int8*** — number
- ***2 byte unsigned integer - dt.uint16*** — number
- ***2 byte signed integer - dt.int16*** — number
- ***2 byte floating point - dt.float16*** — number
- ***3 byte unsigned integer*** – 232.600 RGB color
- ***3 byte time / day - dt.time*** — table with the following items:
  - day — number (0-7)
  - hour — number (0-23)
  - minute — number (0-59)
  - second — number (0-59)
- ***3 byte date - dt.date*** — table with the following items:
  - day — number (1-31)
  - month — number (1-12)
  - year — number (1990-2089)
- ***4 byte unsigned integer - dt.uint32*** — number
- ***4 byte signed integer - dt.int32*** — number
- ***4 byte floating point - dt.float32*** — number
- ***4 byte access control - dt.access*** — number, currently not fully supported

2015 Schneider Electric. All rights reserved.

- ***14 byte ASCII string - dt.string*** — string, null characters ('\0') are discarded during decoding

10.6 Data Storage Functions

storage object provides persistent key-value data storage for user scripts. Only the following LUA data types are supported:

- ***boolean***
- ***number***
- ***string***
- ***table***

storage.set(key, value)

Sets new value for the given key. Old value is overwritten. Returns Boolean as the result and an optional error string.

storage.get(key, default)

Gets value for the given key or returns default value (*nil* if not specified) if key is not found in the data storage.

NOTE: All user scripts share the same data storage. Make sure that the same keys are not used to store different types of data.

Example:

- The following examples show the basic syntax of ***storage.set***. Result will return boolean ***true*** since the passed parameters are correct.

```
result=storage.set('my_stored_value_1', 12.21)
```

- This example will return ***false*** as the result because we are trying to store a function which is not possible.

```
testfn=function(t)
return t * t
end
result =storage.set('my_stored_value_2', testfn)-- this will result in an error
```

- The following examples show the basic syntax of ***storage.get***. Assuming that key value was not found, first call will return ***nil*** while the second call will return number ***0*** which was specified as a default value.

```
result =storage.get('my_stored_value_3')-- returns nil if value is not found
result =storage.get('my_stored_value_3', 0)-- returns 0 if value is not found
```

- When storing tables, make sure to check the returned result type. Assume we have created a storage item with key ***test_object_data***.

```
objectdata={}
```


2015 Schneider Electric. All rights reserved.

```
objectdata.temperature=23.1
objectdata.scene='default'
result =storage.set('test_object_data', objectdata)-- store objectdata variable as 'test_object_data'
```

- Now we are retrieving data from storage. Data type is checked for correctness.

```
objectdata=storage.get('test_object_data')
if type(objectdata)=='table'then
if objectdata.temperature> 24 then
-- do something if temperature level is too high
end
end
```

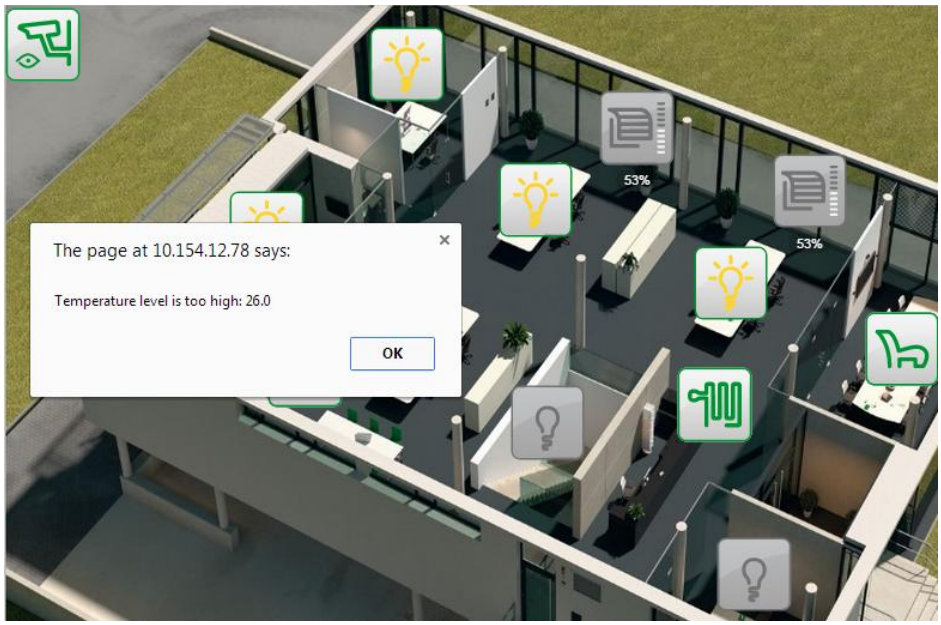
10.7 Alert Functions

Alert (message, [var1, [var2, [var3]]])

Stores alert message and current system time in the main database. All alerts are accessible in the "Alerts" module. This function behaves exactly as LUA ***string.format***.

Example:

```
temperature = 25.3
if temperature > 24 then
-- resulting message: 'Temperature levels are too high: 25.3'
  alert('Temperature level is too high: %.1f', temperature)
end
```



2015 Schneider Electric. All rights reserved.

10.8 Log Functions

Log (var1, [var2, [var3, ...]])

Converts variables to human-readable form and stores them in the main database. All items are accessible in the "Logs" module.

Example:

```
-- log function accepts LUA nil, boolean, number and table (up to 5 nested levels) type variables
a ={ key1 ='value1', key2 =2}
b ='test'
c =123.45
-- logs all passed variables
log(a, b, c)
```

10.9 Time Functions

os.sleep(delay)

Delay the next command execution for the ***delay*** seconds.

os.microtime ()

Returns two values: current timestamp in seconds and timestamp fraction in nanoseconds.

os.udifftime (sec, usec)

Returns time difference as floating point value between now and timestamp components passed to this function (seconds, nanoseconds).

10.10 String Functions

This library provides generic functions for string manipulation, such as finding and extracting substrings, and pattern matching. When indexing a string in LUA, the first character is at position 1 (not at 0, as in C).

Indices are allowed to be negative and are interpreted as indexing backwards, from the end of the string. Thus, the last character is at position -1, and so on.

The string library provides all its functions inside the table string. It also sets a metatable for strings where the __index field points to the string table. Therefore, you can use the string functions in object-oriented style. For instance, *string.byte(s, i)* can be written as *s:byte(i)*. The string library assumes one-byte character encodings.

string.trim (str)

Trims the leading and trailing spaces off a given string.

2015 Schneider Electric. All rights reserved.

string.split (str, sep)

Splits string by given separator string. Returns LUA table.

string.byte (s [, i [, j]])

Returns the internal numerical codes of the characters *s[i], s[i+1], ···, s[j]*. The default value for *i* is 1; the default value for *j* is i.

NOTE: that numerical codes are not necessarily portable across platforms.

string.char (···)

Receives zero or more integers. Returns a string with length equal to the number of arguments, in which each character has the internal numerical code equal to its corresponding argument.

NOTE: Numerical codes are not necessarily portable across platforms.

string.find (s, pattern [, init [, plain]])

Looks for the first match of pattern in the string s. If it finds a match, then find returns the indices of *s* where this occurrence starts and ends; otherwise, it returns ***nil***. A third, optional numerical argument init specifies where to start the search; the default value is 1 and can be negative. A value of true as a fourth, optional argument plain turns off the pattern matching facilities, so the function does a plain "find substring" operation, with no characters in the pattern being considered "magic".

NOTE: If plain is given, then init must be given as well. If the pattern has captures, then in a successful match the captured values are also returned, after the two indices.

string.format (formatstring, ···)

Returns a formatted version of its variable number of arguments following the description given in its first argument (which must be a string). The format string follows the same rules as the printf family of standard C functions. The only differences are that, the options/modifiers *, l, L, n, p, and h are not supported and that there is an extra option, q. The q option formats a string in a form suitable to be safely read back by the LUA interpreter: the string is written between double quotes, and all double quotes, newlines, embedded zeros, and backslashes in the string are correctly escaped when written. For instance, the call:

string.format('%q', 'a string with "quotes" and **\n** new line')

will produce the string:

"a string with \"quotes\" and \

new line"

The options *c, d, E, e, f, g, G, i, o, u, X,* and *x* all expect a number as argument whereas, *q* and *s* expect a string. This function does not accept string values containing embedded zeros, except as arguments to the *q* option.

2015 Schneider Electric. All rights reserved.

string.gmatch (s, pattern)

Returns an iterator function that, each time it is called, returns the next captures from pattern over string s. If the pattern specifies no captures, then the whole match is produced in each call. As an example, the following loop:

```
s = "hello world from LUA"
for w in string.gmatch(s, "%a+") do
  print(w)
end
```

will iterate over all the words from string *s*, printing one per line. The next example collects all pairs *key=value* from the given string into a table:

```
t = {}
s = "from=world, to=LUA"
for k, v in string.gmatch(s, "(%w+)=(%w+)") do
  t[k] = v
end
```

For this function, a '^' at the start of a pattern does not work as an anchor, as this would prevent the iteration.

string.gsub (s, pattern, repl [, n])

Returns a copy of s in which all (or the first n, if given) occurrences of the pattern have been replaced by a replacement string specified by repl, which can be a string, a table, or a function. gsub also returns, as its second value, the total number of matches that occurred.

If ***repl*** is a string, then its value is used for replacement. The character % works as an escape character: any sequence in repl of the form *%n*, with *n* between 1 and 9, stands for the value of the n-th captured substring (see below). The sequence %0 stands for the whole match. The sequence %% stands for a single %.

If ***repl*** is a table, then the table is queried for every match, using the first capture as the key; if the pattern specifies no captures, then the whole match is used as the key.

If ***repl*** is a function, then this function is called every time a match occurs, with all captured substrings passed as arguments, therefore; if the pattern specifies no captures, then the whole match is passed as a sole argument.

If the value returned by the table query or by the function call is a string or a number, then it is used as the replacement string; otherwise, if it is ***false*** or ***nil***, then there is no replacement (that is, the original match is kept in the string).

Example:

```
x = string.gsub("hello world", "(%w+)", "%1 %1")
```

--> x="hello hello world world"

```
x = string.gsub("hello world", "%w+", "%0 %0", 1)
```


2015 Schneider Electric. All rights reserved.

```
    --> x="hello hello world"

x = string.gsub("hello world from LUA", "(%w+)%s*(%w+)", "%2 %1")

    --> x="world hello LUA from"

x = string.gsub("home = $HOME, user = $USER", "%$(%w+)", os.getenv)

    --> x="home = /home/roberto, user = roberto"

x = string.gsub("4+5 = $return 4+5$", "%$(.-)%$", function (s)

    return loadstring(s)()

 end)

    --> x="4+5 = 9"

local t = {name="LUA", version="5.1"}

          x = string.gsub("$name-$version.tar.gz", "%$(%w+)", t)

    --> x="LUA-5.1.tar.gz"
```

string.len (s)

Receives a string and returns its length. The empty string "" has length 0. Embedded zeros are counted, so "a\000bc\000" has length 5.

string.lower (s)

Receives a string and returns a copy of this string with all uppercase letters changed to lowercase. All other characters are left unchanged. The definition of what an uppercase letter, depends on the current locale.

string.match (s, pattern [, init])

Looks for the first match of pattern in the string s. If it finds one, then the match returns the captures from the pattern; otherwise, it returns ***nil***. If the pattern specifies no captures, then the whole match is returned. A third, optional numerical argument init specifies where to start the search; its default value is 1 and can be negative.

string.rep (s, n)

Returns a string that is the concatenation of n copies of the string s.

string.reverse (s)

Returns a string that is the string s reversed.

string.sub (s, i [, j])

Returns the substring of s that starts at i and continues until j; i and j can be negative. If j is absent, then it is assumed to be equal to -1 (which is the same as the string length). In particular, the call *string.sub(s,1,j)* returns a prefix of s with length j, and *string.sub(s, -i)* returns a suffix of *s* with length *i*.

2015 Schneider Electric. All rights reserved.

string.upper (s)

Receives a string and returns a copy of this string with all lowercase letters changed to uppercase. All other characters are left unchanged. The definition of what a lowercase letter is, depends on the current locale.

2015 Schneider Electric. All rights reserved.

Patterns

Character Class:

A character class is used to represent a set of characters. The following combinations are allowed in describing a character class:

- **x**: (where x is not one of the magic characters ^$()%.[]*+-?) represents the character x itself.
- **.**: (a dot) represents all characters.
- **%a**: represents all letters.
- **%c**: represents all control characters.
- **%d**: represents all digits.
- **%l**: represents all lowercase letters.
- **%p**: represents all punctuation characters.
- **%s**: represents all space characters.
- **%u**: represents all uppercase letters.
- **%w**: represents all alphanumeric characters.
- **%x**: represents all hexadecimal digits.
- **%z**: represents the character with representation 0.
- **%x**: (where x is any non-alphanumeric character) represents the character x. This is the standard way to escape the magic characters. Any punctuation character (even the non magic) can be preceded by a '%' when used to represent itself in a pattern.
- **[set]**: represents the class which is the union of all characters in set. A range of characters can be specified by separating the end characters of the range with a '-'. All classes %x described above can also be used as components in set. All other characters in set represent themselves. For example, [%w_] (or [_%w]) represents all alphanumeric characters plus the underscore, [0-7] represents the octal digits, and [0-7%l%-] represents the octal digits plus the lowercase letters plus the '-' character.
- The interaction between ranges and classes are not defined. Therefore, patterns like [%a-z] or [a-%%] have no meaning.
- **[^set]**: represents the complement of set, where set is interpreted as above.

For all classes represented by single letters (%a, %c, etc.), the corresponding uppercase letter represents the complement of the class. For instance, %S represents all non-space characters.

2015 Schneider Electric. All rights reserved.

The definitions of letter, space, and other character groups depend on the current locale. In particular, the class [a-z] may not be equivalent to %l.

Pattern Item:

A pattern item can be:

- a single character class, which matches any single character in the class;
- a single character class followed by '*', which matches 0 or more repetitions of characters in the class. These repetition items will always match the longest possible sequence;
- a single character class followed by '+', which matches 1 or more repetitions of characters in the class. These repetition items will always match the longest possible sequence;
- a single character class followed by '-', which also matches 0 or more repetitions of characters in the class. Unlike '*', these repetition items will always match the shortest possible sequence;
- a single character class followed by '?', which matches 0 or 1 occurrence of a character in the class;
- %n, for n between 1 and 9; such item matches a substring equal to the n-th captured string (see below);
- %bxy, where x and y are two distinct characters; such item matches strings that start with x, end with y, and where the x and y are balanced. Henceforth, if one reads the string from left to right, counting +1 for an x and -1 for a y, the ending y is the first y where, the count reaches 0. For instance, the item %b() matches expressions with balanced parentheses.

Pattern:

A pattern is a sequence of pattern items. A '^' at the beginning of a pattern anchors the match at the beginning of the subject string. A '$' at the end of a pattern anchors the match at the end of the subject string. At other positions, '^' and '$' have no special meaning and represent themselves.

Captures:

A pattern can contain sub-patterns enclosed in parentheses; they describe captures. When a match succeeds, the substrings of the subject string that match captures are stored (captured) for future use. Captures are numbered according to their left parentheses. For instance, in the pattern "(a*(.)%w(%s*))", the part of the string matching "a*(.)%w(%s*)" is stored as the first capture (and therefore, has number 1); the character matching "." is captured with number 2, and the part matching "%s*" has number 3.

As a special case, the empty capture (), captures the current string position (a number). For instance, if we apply the pattern "()aa()" on the string "flaaap", there will be two captures: 3 and 5. A pattern cannot contain embedded zeros. Use %z instead.

2015 Schneider Electric. All rights reserved.

10.11 Input and Output Functions

io.exists (path)
Checks if given path (file or directory) exists. Return boolean.

io.readfile (file)
Reads whole file at once. Return file contents as a string on success or nil on error.

io.writefile (file, data)
Writes given data to a file. Data can be either a value convertible to string or a table of such values. When data is a table, then each table item is terminated by a new line character. Return boolean as write result when file can be open for writing or nil when file cannot be accessed.

10.12 Script Control Function

script.enable('scriptname')
Enable the script with the name ***scriptname***.

script.disable('scriptname')
Disable the script with the name ***scriptname***.

status = script.status('scriptname')
Returns true/false if script is found, nil otherwise.

10.13 Conversions

Compatibility layer: *lmcore* is an alias of *cnv*.

cnv.strtohex (str)
Converts given binary string to a hex-encoded string.

cnv.hextostr (hex [, keepnulls])
Converts given hex-encoded string to a binary string. NULL characters are ignored by default, but can be included by setting second parameter to true.

cnv.tonumber (value)
Converts the given value to number using the following rules: numbers and valid numeric strings are treated as is, boolean *true* is 1, boolean *false* is 0, everything else is *nil*.

cnv.hextoint(hexvalue, bytes)
Converts the given hex string to and integer of a given length in bytes.

cnv.inttohex(intvalue, bytes)
Converts the given integer to a hex string of given bytes.

cnv.strtohex(str)
Converts the given binary string to a hex-encoded string.

2015 Schneider Electric. All rights reserved.

cnv.hextostr(hexstr)
Converts the given hex-encoded string to a binary string.

10.14 Bit Operators

bit.bnot (value)
Binary not

bit.band (x1 [, x2...])
Binary and between any number of variables

bit.bor (x1 [, x2...])
Binary and between any number of variables

bit.bxor (x1 [, x2...])
Binary and between any number of variables

bit.lshift (value, shift)
Left binary shift

bit.rshift (value, shift)
Right binary shift

10.15 Input and Output Facilities

The I/O library provides two different styles for file manipulation. The first one uses implicit file descriptors; that is, there are operations to set a default input file and a default output file, and all input/output operations are over these default files. The second style uses explicit file descriptors.

When using implicit file descriptors, all operations are supplied by table *io*. When using explicit file descriptors, the operation *io.open* returns a file descriptor, and then all the operations are supplied as methods of the file descriptor.

The table *io* also provides three predefined file descriptors with their usual meanings from C: *io.stdin, io.stdout*, and *io.stderr*. The I/O library never closes these files.

Unless otherwise stated, all I/O functions return *nil* on failure (plus an error message as a second result and a system-dependent error code as a third result) and some value different from *nil* on success.

io.close ([file])
Equivalent to *file:close()*. Without a file, closes the default output file.

io.flush ()
Equivalent to file:flush over the default output file.

io.input ([file])
When called with a file name, it opens the named file (in text mode), and sets its handle as the default input file. When called with a file handle, it simply sets this file handle as the default

2015 Schneider Electric. All rights reserved.

input file. When called without parameters, it returns the current default input file. In case of errors this function raises the error, instead of returning an error code.

io.lines ([filename])

Opens the given file name in read mode and returns an iterator function that, each time it is called, returns a new line from the file. Therefore, the construction will iterate over all lines of the file. When the iterator function detects the end of file, it returns nil (to finish the loop) and automatically closes the file.

for line in io.lines(filename) do body end

The call *io.lines()* (with no file name) is equivalent to *io.input():lines()*; that is, it iterates over the lines of the default input file. In this case, it does not close the file when the loop ends.

io.open (filename [, mode])

This function opens a file, in the mode specified in the string mode. It returns a new file handle, or, in case of errors, nil plus an error message. The mode string can be any of the following:

- "r": read mode (the default);
- "w": write mode;
- "a": append mode;
- "r+": update mode, all previous data is preserved;
- "w+": update mode, all previous data is erased;
- "a+": append update mode, previous data is preserved; writing is only allowed at the end of file.

The mode string can also have a 'b' at the end, which is needed in some systems to open the file in binary mode. This string is exactly what is used in the standard C function *fopen*.

io.output ([file])

Similar to io.input, but operates over the default output file.

10.16 Mathematical functions

This library is an interface to the standard C math library. It provides all its functions inside the table math.

math.abs (x)

Returns the absolute value of x.

math.acos (x)

Returns the arc cosine of x (in radians).

math.asin (x)

Returns the arc sine of x (in radians).

2015 Schneider Electric. All rights reserved.

math.atan (x)
Returns the arc tangent of x (in radians).

math.atan2 (y, x)
Returns the arc tangent of y/x (in radians), but uses the signs of both parameters to find the quadrant of the result. (It also handles correctly the case of x being zero.)

math.ceil (x)
Returns the smallest integer larger than or equal to x.

math.cos (x)
Returns the cosine of x (assumed to be in radians).

math.cosh (x)
Returns the hyperbolic cosine of x.

math.deg (x)
Returns the angle x (given in radians) in degrees.

math.exp (x)
Returns the value e^x.

math.floor (x)
Returns the largest integer smaller than or equal to x.

math.fmod (x, y)
Returns the remainder of the division of x by y that rounds the quotient towards zero.

math.frexp (x)
Returns m and e such that x = $m2^e$, e is an integer and the absolute value of m is in the range [0.5, 1) (or zero when x is zero).

math.huge
The value HUGE_VAL, a value larger than or equal to any other numerical value.

math.ldexp (m, e)
Returns $m2^e$, (e should be an integer).

math.log (x)
Returns the natural logarithm of x.

math.log10 (x)
Returns the base-10 logarithm of x.

math.max (x, ···)
Returns the maximum value among its arguments.

math.min (x, ···)
Returns the minimum value among its arguments.

math.modf (x)
Returns two numbers, the integral part of x and the fractional part of x.

2015 Schneider Electric. All rights reserved.

math.pi
The value of pi.

math.pow (x, y)
Returns x^y . (You can also use the expression x^y to compute this value.)

math.rad (x)
Returns the angle x (given in degrees) in radians.

math.random ([m [, n]])
This function is an interface to the simple pseudo-random generator function rand provided by ANSI C. (No guarantees can be given for its statistical properties.)
When called without arguments, returns a uniform pseudo-random real number in the range [0,1). When called with an integer number m, math.random returns a uniform pseudo-random integer in the range [1,m]. When called with two integer numbers m and n, math.random returns a uniform pseudo-random integer in the range [m, n].

math.randomseed (x)
Sets x as the "seed" for the pseudo-random generator: equal seeds produce equal sequences of numbers.

math.sin (x)
Returns the sine of x (assumed to be in radians).

math.sinh (x)
Returns the hyperbolic sine of x.

math.sqrt (x)
Returns the square root of x. (You can also use the expression x^0.5 to compute this value.)

math.tan (x)
Returns the tangent of x (assumed to be in radians).

math.tanh (x)
Returns the hyperbolic tangent of x.

10.17 Table manipulation

This library provides generic functions for table manipulation. It provides all its functions inside the table. Most functions in the table library assume that the table represents an array or a list. For these functions, when we talk about the "length" of a table we mean the result of the length operator.

table.concat (table [, sep [, i [, j]]])
Given an array where all elements are strings or numbers, returns *table[i]..sep..table[i+1]* ··· *sep..table[j]*. The default value for sep is the empty string, the default for *i* is 1, and the default for *j* is the length of the table. If *i* is greater than *j*, it returns the empty string.

2015 Schneider Electric. All rights reserved.

table.insert (table, [pos,] value)

Inserts element value at position pos in table, shifting up other elements to open space, if necessary. The default value for *pos* is *n+1*, where n is the length of the table, so that a call *table.insert(t,x)* inserts x at the end of table t.

table.maxn (table)

Returns the largest positive numerical index of the given table, or zero if the table has no positive numerical indices.(To do its job, this function does a linear traversal of the whole table).

table.remove (table [, pos])

Removes from table the element at position pos, shifting down other elements to close the space, if necessary. Returns the value of the removed element. The default value for pos is n, where n is the length of the table, so that a call *table.remove(t)* removes the last element of table t.

table.sort (table [, comp])

Sorts table elements in a given order, in-place, from *table[1]* to *table[n]*, where n is the length of the table. If comp is given, then it must be a function that receives two table elements, and returns true when the first is less than the second (so that not *comp(a[i+1],a[i]*) will be true after the sort). If comp is not given, then the standard LUA operator < is used instead.

The sort algorithm is not stable; that is, elements considered equal by the given order may have their relative positions changed by the sort.

10.18 Operating System Facilities

os.date ([format [, time]])

Returns a string or a table containing date and time, formatted according to the given string format. If the time argument is present, this is the time to be formatted (see the *os.time* function for a description of this value). Otherwise, date formats the current time.

If format starts with '!', then the date is formatted in Coordinated Universal Time. After this optional character, if format is the string "*t", then date returns a table with the following fields: year (four digits), month (1--12), day (1--31), hour (0--23), min (0--59), sec (0--61), wday (weekday, Sunday is 1), yday (day of the year), and isdst (daylight saving flag, a boolean).

If format is not "*t", then date returns the date as a string, formatted according to the same rules as the C function strftime.

When called without arguments, date returns a reasonable date and time representation that depends on the host system and on the current locale (that is, *os.date()* is equivalent to os.date("%c")).

os.difftime (t2, t1)

Returns the number of seconds from time t1 to time t2. In POSIX, Windows, and some other systems, this value is exactly t2-t1.

 2015 Schneider Electric. All rights reserved.

os.execute ([command])

This function is equivalent to the C function system. It passes command to be executed by an operating system shell. It returns a status code, which is system-dependent. If the command is absent, then it returns nonzero if a shell is available and zero otherwise.

os.exit ([code])

Calls the C function exit, with an optional code, to terminate the host program. The default value for code is the success code.

os.getenv (varname)

Returns the value of the process environment variable varname, or ***nil*** if the variable is not defined.

os.remove (filename)

Deletes the file or directory with the given name. Directories must be empty to be removed. If this function fails, it returns nil, plus a string describing the error.

os.rename (oldname, newname)

Renames file or directory named oldname to newname. If this function fails, it returns *nil*, plus a string describing the error.

os.time ([table])

Returns the current time when called without arguments, or a time representing the date and time specified by the given table. This table must have fields year, month, and day, and may have fields hour, min, sec, and *isdst* (for a description of these fields, see the *os.date* function).

The returned value is a number, whose meaning depends on your system. In POSIX, Windows, and some other systems, this number counts the number of seconds since some given start time (the "epoch"). In other systems, the meaning is not specified, and the number returned by time can be used only as an argument to date and *difftime*.

os.tmpname ()

Returns a string with a file name that can be used for a temporary file. The file must be explicitly opened before its use and explicitly removed when no longer needed. On some systems (POSIX), this function also creates a file with that name, to avoid security risks. (Someone else might create the file with wrong permissions in the time between getting the name and creating the file.) You still have to open the file to use it and to remove it (even if you do not use it).

When possible, you may prefer to use *io.tmpfile*, which automatically removes the file when the program ends

10.19 Extended Function Library

toboolean(value)

Converts the given value to boolean using the following rules: *nil*, boolean *false*, *0*, *empty* string, *'0'* string are treated as *false*, everything else as *true.*

string.split(str, sep)

Splits the given string into chunks by the given separator. Returns LUA table.

 2015 Schneider Electric. All rights reserved.

knxlib.decodeia(indaddressa, indaddressb)

Converts the binary-encoded individual address to LUA string. This function accepts either one or two arguments (interpreted as two single bytes).

knxlib.decodega(groupaddressa, groupaddressb)

Converts the binary-encoded group address to LUA string. This function accepts either one or two arguments (interpreted as two single bytes).

knxlib.encodega(groupaddress, separate)

Converts the LUA string to binary-encoded group address. Returns group address a single LUA number when second argument is *nil* or *false* and two separate bytes otherwise.

ipairs (t)

Returns three values: an iterator function, the table t, and 0, so that the construction will iterate over the pairs (1,t[1]), (2,t[2]), ···, up to the first integer key absent from the table.

for i,v in ipairs(t) do *body* end

next (table [, index])

Allows a program to traverse all fields of a table. The first argument is a table, and the second argument is an index in this table. Next returns the next index of the table and its associated value. When called with *nil* as its second argument, next returns an initial index and its associated value. When called with the last index, or with *nil* in an empty table, next returns *nil*. If the second argument is absent, then it is interpreted as *nil*. In particular, you can use *next(t*) to check whether a table is empty. The order in which the indices are enumerated is not specified, even for numeric indices. (To traverse a table in numeric order, use a numerical for or the *ipairs* function.) The behavior of next is undefined if, during the traversal, you assign any value to a non-existent field in the table. You may however modify the existing fields. In particular, you may clear existing fields.

pairs (t)

Returns the three values: the *next* function, the table *t*, and *nil*, so that the construction will iterate over all key–value pairs of table t.

for k,v in pairs(t) do body end

tonumber (e [, base])

Tries to convert its argument to a number. If the argument is already a number or a string convertible to a number, then tonumber returns this number; otherwise, it returns *nil*.

An optional argument specifies the base to interpret the numeral. The base may be any integer between 2 and 36, inclusive. In bases above 10, the letter 'A' (in either upper or lower case) represents 10, 'B' represents 11, and so forth, with 'Z' representing 35. In base 10 (the default), the number can have a decimal part, as well as an optional exponent part. In other bases, only unsigned integers are accepted.

tostring (e)

Receives an argument of any type and converts it to a string in a reasonable format. For complete control of how numbers are converted, use *string.format*.

If the metatable of e has a "__tostring" field, then *tostring* calls the corresponding value with e as argument, and uses the result of the call as its result.

 2015 Schneider Electric. All rights reserved.

type (v)

Returns the type of its only argument, coded as a string. The possible results of this function are "nil" (a string, not the value *nil*), "number", "string", "boolean", "table", "function", "thread", and "userdata".

2015 Schneider Electric. All rights reserved.

11 Script Examples

11.1 Binary Filter

Create two 1-bit group addresses under Object tab where:

1/1/1 input

1/1/2 output

Create event–based script and attach it to group 1/1/1. Script will run each time group 1/1/1 receive telegram.

Add the following code to the Script editor:

```
value_1 = grp.getvalue('1/1/1')
if value_1 == true then
-- do nothing
elseif value_1 == false then
grp.write('1/1/2', false)
end
```

11.2 Binary Gate with Bit Gate

Create three 1-bit group addresses under Object tab where:

1/1/1 input

1/1/2 gate

1/1/3 output

Create event –based script and attach it to group 1/1/1. Script will run each time group 1/1/1 receive telegram.

Add the following code to the Script editor:

```
value_1 = grp.getvalue('1/1/1') --input
value_2 = grp.getvalue('1/1/2') --gate
if value_2 == true then
-- do nothing
elseif value_2 == false then
grp.write('1/1/3', value_1) --output
end
```


2015 Schneider Electric. All rights reserved.

11.3 Gate with Byte Gate

Create three group addresses under Object tab where:

1/1/1 input – any type but the same as output

1/1/2 gate- byte object

1/1/3 output – the same as input

Create event–based script and attach it to group 1/1/1. Script will run each time group 1/1/1 receive telegram.

Add the following code to the Script editor:

```
value_1 = grp.getvalue('1/1/1') -- input
value_2 = grp.getvalue('1/1/2') --gate
if value_2 == 0 then
-- do nothing
elseif  value_2 < 0 or value_2 > 0   then
grp.write('1/1/3', value_1) --output
end
```

11.4 Or - Port (2 in 1 0ut)

Create three 1-bit group addresses under Object tab where:

1/1/1 value 1

1/1/2 value 2

1/1/3 output

Add tag OR1 to value1 and value2 group addresses.

Create event–based script and attach it to Tag OR1. Script will run each time group 1/1/1 or group 1/1/2 receive telegram.

Add the following code to the Script editor:

```
value_1 = grp.getvalue('1/1/1')
value_2 = grp.getvalue('1/1/2')
if value_1 == true or value_2 == true then
grp.write('1/1/3', true)
else
grp.write('1/1/3', false)
end
```


2015 Schneider Electric. All rights reserved.

11.5 And - Port (2 in 1 0ut)

Create three 1-bit group addresses under Object tab where:

1/1/1 value 1

1/1/2 value 2

1/1/3 output

Add tag AND1 to value1 and value2 group addresses.

Create event–based script and attach it to Tag AND1. Script will run each time group 1/1/1 or group 1/1/2 receive telegram.

Add the following code to the Script editor:

```
value_1 = grp.getvalue('1/1/1')
value_2 = grp.getvalue('1/1/2')
if value_1 == true and value_2 == true then
grp.write('1/1/3', true)
else
grp.write('1/1/3', false)
end
```

11.6 Or - Port (5 in 2 0ut)

Create group addresses under Object tab where:

1/1/1 value 1 - 1bit

1/1/2 value 2 - 1bit

1/1/3 value 3 - 1bit

1/1/4 value 4 - 1bit

1/1/5 value 5 - 1bit

1/1/6 bit_output - 1bit

1/1/7 byte_output - 1byte

Add tag OR2 to group addresses value1, value2, value3, value4 and value 5.

Create event–based script and attach it to Tag OR2. Script will run each time groups 1/1/1, 1/1/2, 1/1/3, 1/1/4, 1/1/5 receive telegram

Add the following code to the Script editor:

2015 Schneider Electric. All rights reserved.

```
value_1 = grp.getvalue('1/1/1')
value_2 = grp.getvalue('1/1/2')
value_3 = grp.getvalue('1/1/3')
value_4 = grp.getvalue('1/1/4')
value_5 = grp.getvalue('1/1/5')
if value_1 == true or value_2 == true or value_3 == true or value_4 == true or value_5 == true then
grp.write('1/1/6', true) -- bit to 1
grp.write('1/1/7', 255) -- byte to 255
else
grp.write('1/1/6', false) -- bit to 0
grp.write('1/1/7', 0) -- byte to 0
end
```

11.7 And - Port (5 in 2 0ut)

Create group addresses under Object tab where:

1/1/1 value 1 - 1bit
1/1/2 value 2 - 1bit
1/1/3 value 3 - 1bit
1/1/4 value 4 - 1bit
1/1/5 value 5 - 1bit
1/1/6 bit_output - 1bit
1/1/7 byte_output - 1byte

Add tag AND2 to group addresses value1, value2, value3, value4 and value 5.

Create event–based script and attach it to Tag AND2. Script will run each time groups 1/1/1, 1/1/2, 1/1/3, 1/1/4, 1/1/5 receive telegram

Add the following code to the Script editor:

```
value_1 = grp.getvalue('1/1/1')
value_2 = grp.getvalue('1/1/2')
value_3 = grp.getvalue('1/1/3')
value_4 = grp.getvalue('1/1/4')
value_5 = grp.getvalue('1/1/5')
if value_1 == true and value_2 == true and value_3 == true and value_4 == true and value_5 == true
then
grp.write('1/1/6', true) -- bit to 1
grp.write('1/1/7', 255) -- byte to 255
else
grp.write('1/1/6', false) -- bit to 0
grp.write('1/1/7', 0) -- byte to 0
end
```


2015 Schneider Electric. All rights reserved.

11.8 Telegram Transformer (0/1 bit to 0-255 byte)

Create two group addresses under Object tab where:

1/1/1 input – 1-bit
1/1/2 output – 1-byte

Create event–based script and attach it to group 1/1/1. Script will run each time group 1/1/1 receive telegram.

Add the following code to the Script editor:

```
value_1 = grp.getvalue('1/1/1')
if value_1 == true then -- bit value (in)
grp.write('1/1/2', 255) -- byte value (out)
else
grp.write('1/1/2', 0) -- byte value (out)
end
```

11.9 Compare Value

```
value_1 = grp.getvalue('1/1/1')
value_2 = grp.getvalue('1/1/2')
if value_1 == value_2 then
grp.write('1/1/3', true) -- bit to 1
grp.write('1/1/4', 255) -- byte to 255
else
grp.write('1/1/3', false) -- bit to 0
grp.write('1/1/4', 0) -- byte to 0
end
```

11.10 Save Scene 1 (RGB value)

```
value_1 = grp.getvalue('1/1/1') --RED
value_2 = grp.getvalue('1/1/2') --GREEN
value_3 = grp.getvalue('1/1/3') --BLUE
storage.set('Scene1_Red', value_1)
storage.set('Scene1_Green', value_2)
storage.set('Scene1_Blue', value_3)
```


2015 Schneider Electric. All rights reserved.

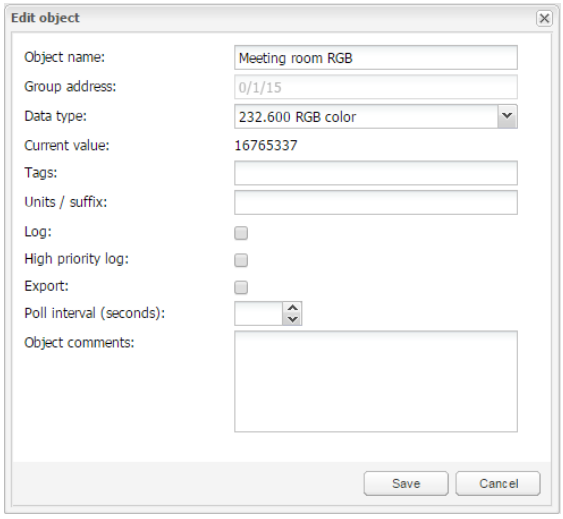
11.11 Call Scene 1 (RGB value)

```
value_1 = storage.get('Scene1_Red')
value_2 = storage.get('Scene1_Green')
value_3 = storage.get('Scene1_Blue')
if not value_1 then
--if storage value does not exist do nothing
else
grp.write('1/1/1', value_1) --RED
end
if not value_2 then
--if storage value does not exist do nothing
else
grp.write('1/1/2', value_2) --GREEN
end
if not value_3 then
--if storage value does not exist do nothing
else
grp.write('1/1/3', value_3) --BLUE
end
```

11.12 RGB object

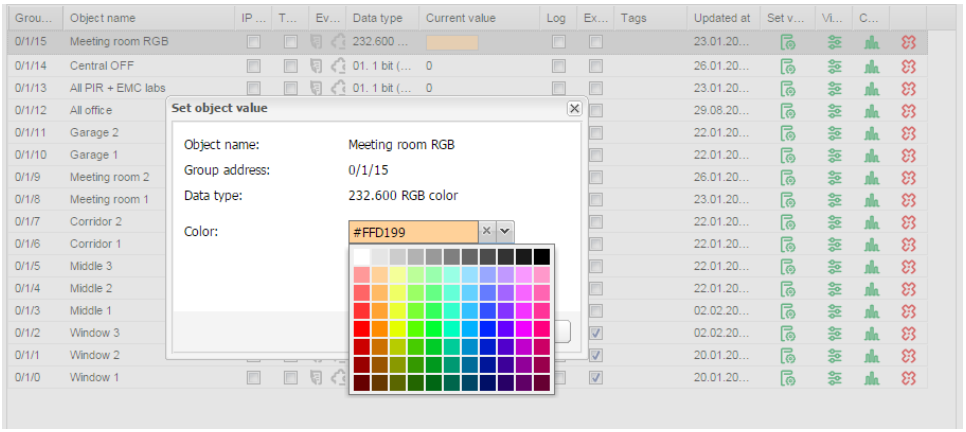
11.12.1 How to operate the RGB object

- Create object and define *Object parameters* in *Configurator/Objects/double click on the object*



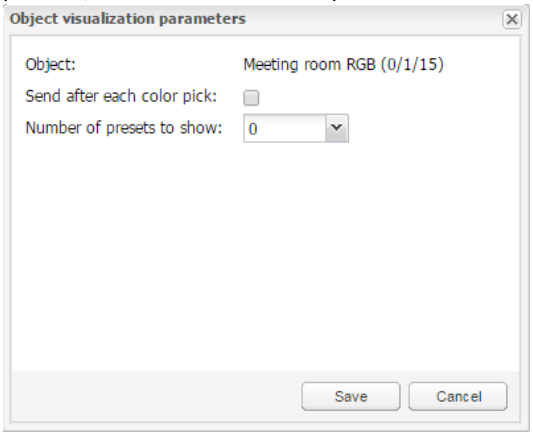
2015 Schneider Electric. All rights reserved.

- Set RGB color in *Configurator/Objects/Set value*

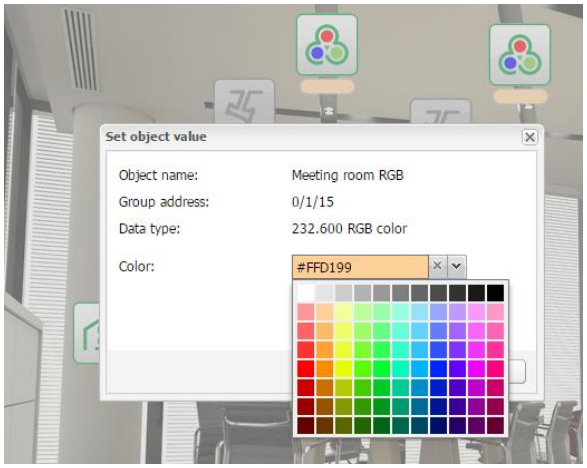


- Set Object visualization parameters in *Configurator/Objects/Vis. Params*

If the option *Send after each color pick* is ticked, a new updated object with selected color will be automatically sent to bus after releasing the left mouse button (PC) or release finger (touch screen) in the Visualization screen
(see *PC/Tablet Visualization*)

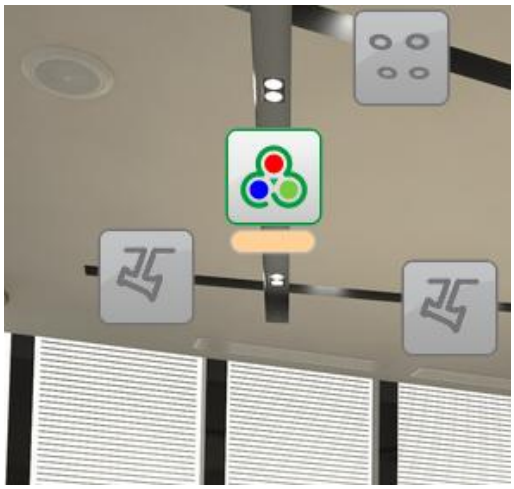


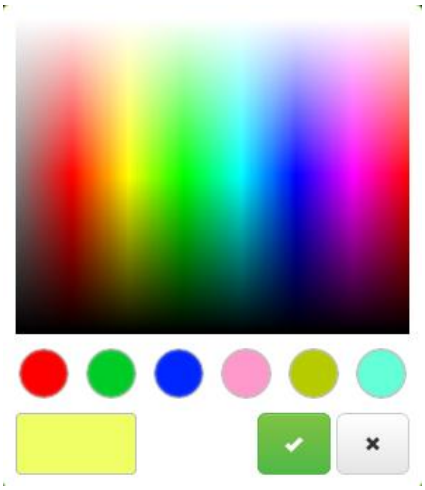
- Set RGB color in *Configurator/Visualization* (identical to *Configurator/Objects/Set value)*



2015 Schneider Electric. All rights reserved.

- Set color as a User in *PC/Tablet Visualization*





11.12.2 Script for controlling RGB LED color

```
---------------------------------------------------------------------------------------------------------
-- This script is splitting  1 x 3byte RGB object in to 3 x 1byte value objects
-- Create event based script which will run from the 3 byte RGB object
-- Fill configurable parameters with group addresses or groups names
-- RGB and 1 byte objects must have correct data type

------------------- Configurable parameters ----------------------------------------------------------

redGroup   = '1/1/5' --- modify ether group address or name of group
greenGroup = 'LED1 Green Value' --- modify ether group address or name of group
blueGroup  = '1/1/7' --- modify ether group address or name of group
---------------------------------------------------------------------------------------------------------

value = event.getvalue()

Blue= bit.band(value, 0xFF)
Green = bit.rshift(bit.band(value, 0xFF00), 8)
Red = bit.rshift(bit.band(value, 0xFF0000), 16)

grp.write(redGroup, Red, dt.uint8)
grp.write(greenGroup, Green, dt.uint8)
grp.write(blueGroup, Blue, dt.uint8)
This script splits one 3 byte RGB object into three 1 byte value objects what control the single
reg/green/blue colors.#
```

11.12.3 Script for reading status from the RGB LED

```
---------------------------------------------------------------------------------------------------------
-- This script is converting 3 x 1byte status objects in to 1 x 3byte RGB object
-- Create event based script which will run from a unique TAG, this tag must be
-- attached to red, green, blue 1 byte status objects
```


2015 Schneider Electric. All rights reserved.

```
-- Fill configurable parameters with group addresses or groups names
-- It is recommended to use 1 byte status object for the inputs
-- RGB object must have correct data type
-------------------------------------------------------------------------------------------------------

------------------- Configurable parameters ---------------------------------------------------------

redGroup   = '1/1/1' --- modify ether group address or name of group
greenGroup = 'LED1 Green Status' --- modify ether group address or name of group
blueGroup  = '1/1/3' --- modify ether group address or name of group

rgbGroup   = 'RGB Value' --- modify ether group address or name of group
-------------------------------------------------------------------------------------------------------

red   = grp.find(redGroup)
green = grp.find(greenGroup)
blue  = grp.find(blueGroup)

redHex = red.datahex
greenHex = green.datahex
blueHex = blue.datahex

RGB = lmcore.hextoint(redHex..greenHex..blueHex)

grp.write(rgbGroup, RGB)
```

11.13 Hysteresis

```
(do not change object 1/1/2 when value of object 1/1/1 is between 100 and 200)
value_1 = grp.getvalue('1/1/1') -- byte value
if value_1 < 100 then
grp.write('1/1/2', false) -- bit to 0

elseif value_1 > 200 then
grp.write('1/1/2', true) -- bit to 0
end
```

11.14 Random byte value

```
steps = 255 -- possible steps change this value to lower value to make bigger steps
random = math.random(0, (steps - 1)) * 255 / (steps - 1)
outcome = (math.floor(random))
value_1 = grp.getvalue('1/1/1')
grp.write('1/1/1', outcome) -- Write random byte value to object
```


2015 Schneider Electric. All rights reserved.

11.15 Cyclic Repeater (delay 60 seconds)

```
value_1 = grp.getvalue('1/1/1')
if value_1 == true then
repeat
value_1 = grp.getvalue('1/1/1')
if value_1 == true then
grp.write('1/1/2', true)
-- wait for 60 seconds
os.sleep(60)
end
until value_1 == false
end
```

11.16 Stepper / Counter Positive input

```
value_1 = grp.getvalue('1/1/1') -- Positive input
if value_1 == true then
Stepper_Value = storage.get('Value_Stepper_1')
if not Stepper_Value then
Stepper_Value = 0
end
if Stepper_Value == 255 then
else
Stepper_Value = Stepper_Value + 1
end
storage.set('Value_Stepper_1', Stepper_Value)
grp.write('1/1/4', Stepper_Value)
end
```

11.17 Stepper / Counter Negative input

```
value_1 = grp.getvalue('1/1/2') -- Negative input
if value_1 == true then
Stepper_Value = storage.get('Value_Stepper_1')
if not Stepper_Value then
Stepper_Value = 0
end
if Stepper_Value == 0 then
else
Stepper_Value = Stepper_Value - 1
end
storage.set('Value_Stepper_1', Stepper_Value)
grp.write('1/1/4', Stepper_Value)
end
```


2015 Schneider Electric. All rights reserved.

11.18 Reset Stepper / Counter

```
value_1 = grp.getvalue('1/1/3')
if value_1 == true then
storage.set('Value_Stepper_1', 0)
grp.write('1/1/4', 0)
end
```

11.19 On Delay (button set to 'update only internal')

```
value_1 = grp.getvalue('1/1/1')
if value_1 == true then
os.sleep(3) -- Delay time
grp.write('1/1/1', true)
end
```

11.20 Average

```
value_1 = grp.getvalue('1/1/1')
value_2 = grp.getvalue('1/1/2')
Average = value_1 + value_2
Average = (Average / 2)
value_3 = grp.getvalue('1/1/3')
grp.write('1/1/3', Average)
```

11.21 Off Delay

```
value_1 = grp.getvalue('1/1/1')
if value_1 == true then
os.sleep(3) -- Delay time
grp.write('1/1/1', false)
end
```

11.22 Stare Case Timer (with variable time object)

```
value_1 = grp.getvalue('1/1/1')
value_2 = grp.getvalue('1/1/2') -- Variable value
if value_1 == true then
os.sleep(value_2)
grp.write('1/1/1', false)
end
```


2015 Schneider Electric. All rights reserved.

11.23 Value Memory (write to storage)

```
value_1 = grp.getvalue('1/1/1')
storage.set('Storage_Value_Memory_1', value_1)
```

11.24 Value Memory (get from storage)

```
Value_Memory_1 = storage.get('Storage_Value_Memory_1')
if not Value_Memory_1 then
-- do nothing
else
grp.write('1/1/1', Value_Memory_1)
end
```

11.25 Multiplexer (1 in / 3 out)

NOTE: Object type needs to be the same.

```
value_1 = grp.getvalue('1/1/1')
grp.write('1/1/2', Value_1)
grp.write('1/1/3', Value_1)
grp.write('1/1/4', Value_1)
```

11.26 Round Function Using Common Functions

Add following code to common functions

```
-- Rounds a number to the given number of decimal places...
function round(num, idp)
local mult = 10^(idp or 0)
return math.floor(num * mult + 0.5) / mult
end
```

Create script in script editor

```
-- Round function (with global function)
value_1 = grp.getvalue('1/1/1')
round(value_1, 2)  -- using function round from common functions
grp.write('1/1/1', Value_2)
```


2015 Schneider Electric. All rights reserved.

11.27 Write Data and Time to KNX Group Addresses

```
-- get current data as table
now = os.date('*t')
-- system week day starts from sunday, convert it to knx format
wday = now.wday == 1 and 7 or now.wday - 1
-- time table
time = {
day = wday,
hour = now.hour,
minute = now.min,
second = now.sec,
}
-- date table
date = {
day = now.day,
month = now.month,
year = now.year,
}
-- write to bus
grp.write('1/1/2', time, dt.time)
grp.write('1/1/1', date, dt.date)
```

11.28 Write Data to Groups with Tags

Create few 1-bit group addresses and add tag 'Light' to them
Create one more group different one from the others to trigger script.

1/1/1 – Lihgt1 – Tag 'Light'
1/1/2 – Lihgt2 – Tag 'Light'
1/1/3 – Lihgt3 – Tag 'Light'
1/1/4 – Lihgt4 – Tag 'Light'
1/1/5 – Lihgt5 – Tag 'Light'
1/1/6 – Lihgt6 – Tag 'Light'

1/1/10 – Scene active group – no tag attached!

Create event–based script and attach it to group 1/1/10. Script will run each time group 1/1/10 receive telegram.

Add the following code to the Script editor:

```
AllLights = grp.tag('Light')
AllLights: write(true)
```

All lights will be switched on each time group 1/1/10 receive telegram.

2015 Schneider Electric. All rights reserved.

NOTE: Do not start the script from the same tag or group addresses containing the same tag. This will create an infinite loop, which will generate traffic on a bus and high load on processor.
If infinite loop is created, stop the script and reboot homeLYnk.

2015 Schneider Electric. All rights reserved.

12 Application Note Documents

Detailed description of selected topics with focus on easy implementation can be found in application notes documents. The list of application notes documents is continuously updated.

12.1.1 Actual List of Released Application Notes:

- AN001_homeLYnk integration using BACnet_v1.4
- AN003_Modbus power meters and homeLYnk_v1.3
- AN006_Advance techniques in visualization (homeLYnk)_v1.2
- AN007_How to replace micro SD card in homeLYnk
- AN010_RS232 control with homeLYnk
- AN011_Email, SMS and FTP in homeLYnk
- AN012_homeLYnk and SONOS
- AN013_Fetch weather forecast to homeLYnk
- AN015_homeLYnk visualization shortcut on the desktop
- AN017_Addressable limits of the homeLYnk controller_v1.1
- AN020_Philips Hue lamp controlled from homeLYnk
- AN024_Block functions for spaceLYnk and homeLYnk controller

12.1.2 Application Notes Availability

All application note documents can be downloaded from Schneider Electric website.

It is possible to use search field on the top of Schneider Electric webpage in order to find the requested document.

2015 Schneider Electric. All rights reserved.